ІНФОРМАЦІЙНО-УПРАВЛЯЮЧІ СИСТЕМИ ТА ОРГАНІЗАЦІЇ ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ

Навчальний посібник

Харків – 2015

ІНФОРМАЦІЙНО-УПРАВЛЯЮЧІ СИСТЕМИ ТА ОРГАНІЗАЦІЇ ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ

Навчальний посібник

За редакцією С.В. Лістрового

Харків – 2015

УДК 681.3
ББК 32.841
Л 43

Рекомендовано вченою радою Українського державного університету залізничного транспорту як навчальний посібник (витяг з протоколу № 4 від 26 травня 2015 р.)

Рецензенти:

професори С.І. Приходько (УкрДУЗТ),
О.О. Можаєв (НТУ «ХПІ»)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

УКРАЇНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЗАЛІЗНИЧНОГО ТРАНСПОРТУ

Л 43 Лістровий С.В., Лістрова О.С., Мірошник М.А. Інформаційно-управляючі системи та організації паралельних обчислень: Навч. посібник / За ред. С.В. Лістрового. – Харків: УкрДУЗТ, 2015. – 322 с., рис. 139, табл. 19.
ISBN 978-617-654-035-9

Книга являє собою навчальний посібник по курсу застосування теорії алгоритмів і теорії графів до синтезу розподілених систем і мереж. Велику увагу приділено питанням організації управління в телекомунікаційних системах і мережах, а також організації паралельних обчислень при розв’язанні задач дискретної оптимізації на основі теорії графів. Книга буде корисна студентам і фахівцям у галузі комп'ютерних систем та програмування.

Навчальний посібник призначений для студентів вищих навчальних закладів за спеціальностями: спеціалізовані комп’ютерні системи та комп'ютерні інформаційно-управляючі системи.

УДК 681.3
ББК 32.841

ISBN 978-617-654-035-9

Навчальний посібник

Лістровий Сергій Володимирович,
Лістрова Олена Сергіївна,
Мірошник Марина Анатоліївна

ІНФОРМАЦІЙНО-УПРАВЛЯЮЧІ СИСТЕМИ
ТА ОРГАНІЗАЦІЇ ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ

Відповідальний за випуск Мірошник М.А.

Редактор Ібрагімова Н.В.

Підписано до друку 11.06.14 р.

Формат паперу 60х84 1/16. Папір писальний.

Умовн.-друк.арк. 17,50. Тираж 300. Замовлення №

Видавець та виготовлювач Українська державна академія залізничного транспорту,
61050, Харків-50, майдан Фейєрбаха, 7.
Свідоцтво суб’єкта видавничої справи ДК № 2874 від 12.06.2007 р.

УКРАЇНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЗАЛІЗНИЧНОГО ТРАНСПОРТУ

ФАКУЛЬТЕТ АВТОМАТИКИ, ТЕЛЕМЕХАНІКИ ТА ЗВ'ЯЗКУ

Кафедра спеціалізованих комп'ютерних систем

ІНФОРМАЦІЙНО-УПРАВЛЯЮЧІ СИСТЕМИ ТА ОРГАНІЗАЦІЇ ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ

ЗА РЕДАКЦІЄЮ ЛІСТРОВОГО С.В.

НАВЧАЛЬНИЙ ПОСІБНИК

Харків 2015

УДК 004.67:___
Д31
ББК _____

Авторський колектив: **Сергій Володимирович Лістровий, Олена Сергіївна Лістрова, Марина Анатоліївна Мірошник**

Д31 Інформаційно-управляючі системи та організації паралельних обчислень: Навч. посібник / С.В. Лістровий, О.С. Лістрова, М.А. Мірошник; за ред. Лістрового С.В. – Харків: УкрДУЗТ, 2015. – 289 с.

ISBN

Книга являє собою навчальний посібник по курсу застосування теорії алгоритмів і теорії графів до синтезу розподілених систем і мереж. Велику увагу приділено питанням організації управління в телекомунікаційних системах і мережах, а також організації паралельних обчислень при розв’язанні задач дискретної оптимізації на основі теорії графів. Книга буде корисна студентам і фахівцям у галузі комп'ютерних систем та програмування.

Навчальний посібник призначений для студентів вищих навчальних закладів за спеціальностями: спеціалізовані комп’ютерні системи та комп'ютерні інформаційно-управляючі системи.

Іл. 139, табл., 19, бібліогр.: 369 назв.

Затверджено до друку вченою радою Українського державного університету залізничного транспорту. Протокол №__ від __ _____ 20__р.

Рецензенти:

професори Приходько С.І. (УкрДУЗТ),
Можаєв О.О. (НТУ «ХПИ»)

УДК 004.67:_____
ББК ___

ISBN

ЗМІСТ

ПЕРЕДМОВА

Ця книга розповідає про сучасні методи теорії дискретної оптимізації і теорії графів і їх додатки до розв’язань задач управління в розподілених системах і телекомунікаційних мережах на основі задач TMN технологій, з метою забезпечення високої оперативності розв’язання.

Для нових рангових методів у задачах дискретної оптимізації і теорії графів, які в сучасній літературі розкидані по окремих статтях і монографіях, наведено систематичний виклад, надано багато прикладів.

Велику увагу в книзі приділено організації паралельних обчислень при розв’язанні задач у телекомунікаційних системах і мережах.

У розд. 1 розглянуто питання управління телекомунікаційними системами та мережами на основі TMN технологій і математичні моделі задач управління.

У розд. 2 автори намагалися доступно викласти основні поняття теорії алгоритмів і графів, необхідні для аналізу та синтезу систем і мереж.

У розд. 3 розглянуто задачі динамічного управління телекомунікаційними системами та мережами моделями, які є задачами лінійного булевого програмування, і рангові методи їх розв’язання, що дозволяють забезпечувати в телекомунікаційних системах та мережах у реальному часі.

У розд. 4 розглянуто загальну схему розв'язання задач комбінаторної оптимізації і теорії графів і показана можливість її застосування для розв’язання як лінійних, так і нелінійних задач булевого програмування і теорії графів.

У розд. 5 розглянуто питання організації паралельних обчислень при вирішенні завдань телекомунікаційними системами та мережами на основі використання циклічних паралельних структур і показано можливість їх застосування до визначення оптимальних маршрутів передачі інформації в телекомунікаційних системах і мережах.

Автори заздалегідь приносять вибачення за помилки, які можуть бути в даній книзі, що має дуже великий обсяг, і вдячні за відомості про виявлені помилки.

ОСНОВНІ СКОРОЧЕННЯ

АСУП – автоматизована система управління підприємством, ще нижче лежать рівні 15
АСУТП – автоматизованої системи управління технологічними процесами 15
БВН – блок вибору напрямку передачі інформації 257
БД – бази даних 12, 241
БЗ – булеві змінні 87, 88, 89, 90, 91, 93, 97, 99, 116, 124, 129, 131, 132, 134, 135, 139, 154, 155, 157, 169, 170, 179, 180, 189, 201, 206, 207, 264, 266,
БП – блок пам'яті 300
БПС – багатопроцесорна система 195, 196
БР – блок регістрів 193, 268
ВР – вірогідність результатів 200, 203, 204, 105
ДМТ – Детермінована машина Тюрінга 41
ДП – динамічне програмування 88, 89, 169
ЕМВВС – еталонна модель взаємодії відкритих систем 17
ЕО – елементарні операції 157, 160, 161, 176, 179
ЕОК – електронно-обчислювальні комплекси 204, 205, 206
ЕОМ – електронно-обчислювальні машини 18, 38, 40, 41, 58, 87, 91, 158, 173, 203, 243, 246, 255, 267, 269
ЗЗП – зовнішні запам'ятовувальні пристрої 173
ЗНП – задача про найменші покриття 63, 144, 146, 151, 152, 153, 154, 157, 173, 175, 177, 203
ЗНР – задача найменшого розбиття 154, 155, 173
ІРЗ – інформаційно-розрахункові задачі 22, 23, 24, 25, 26, 186, 197, 200, 204, 205, 206, 207, 208, 209, 214
К – комутатор 64, 81, 186, 199
КЗ – канал зв’язку 213, 279, 281
ЛЕ – локальний екстремум 286
МДК – матриці досяжності і контр досяжності 51
МГГ – метод гілок і границь 173, 174
МКР – модель керованого ресурсу 17
НД – багатопроцесорні обчислювальні системи 22, 23
НОД– найкоротше остовне дерево графа 198, 200
НШ – найкоротший шлях 74, 75, 76, 77, 78, 81, 144, 201, 210
ОВМ – обчислювальні вузли мережі 213
ОЗП – оперативна пам'ять 173
ОС – операційна система 12,
ОСМ – операційної системи мережі 19
ПАВ – послідовний аналіз варіантів 89
ПЗ – програмне забезпечення 209

ВСТУП

Аналіз сучасних розподілених систем на основі застосування мереж, проведений у роботах [1-31, 33-37], показує, що останнім часом істотно зросла роль факторів, які визначаються часом, що витрачається системою на доведення інформації про стан керованого процесу до пунктів на обробку, що надходить, і прийняття рішення на основі прийнятої інформації і доведення прийнятого рішення до виконавчих органів.

Сучасні складні системи характеризуються винятковою складністю умов, у яких здійснюється управління. До них можна віднести значний обсяг раптово виникаючих завдань; жорсткий ліміт часу, відведеного на прийняття (уточнення) рішення щодо їх виконання; велику інтенсивність інформаційних потоків між різними ланками управління; високий динамізм зміни обстановки; обмеженість ресурсу, призначеного для розв’язання задач [224-235]. Існують об'єкти, у яких час доведення інформації про стан керованого процесу до пунктів складає одиниці секунд. У таких системах час є найважливішим параметром, бо дуже часто потребує формування керуючих дій у реальному часі.

Всі ланки системи знаходяться під фізичними впливами зовнішнього середовища, що призводять до зміни стану системи. Крім зовнішніх фізичних впливів у системі, що призводять до погіршення її стану, існують внутрішні фізичні зв'язки, які можуть поліпшити її стан за рахунок засобів відновлення.

У загальному випадку всі ланки системи з'єднані прямими і зворотними інформаційними зв'язками, призначеними для передачі команд, донесень про їх виконання і стан керованих об'єктів. Зараз з'являються системи, які належать до класу динамічних інформаційних недетермінованих систем. Їх динамічність полягає як у змінах структури системи, так і її параметрів під впливом зовнішніх умов. Функціонування таких систем здійснюється за допомогою передачі інформації, кількість якої не впливає однозначно на результат управління.

Для розв’язання кожної задачі потрібен певний обсяг інформації. Збільшення або зменшення кількості даних не призводить до однозначних змін ефективності прийнятих рішень і витрат часу. Механізм дії цього закону диктує необхідність конкретного вирішення різних питань удосконалення техніки і технології. Ефективність у мережах, як показано в роботах [1-31, 33-37], залежить від оперативності розв’язання задач динамічного управління потоками інформації, математичними моделями яких є широкий клас задач лінійного булевого програмування, що належать до NP-повних завдань, і класу задач нелінійного булевого програмування, для яких ефективні методи розв’язання практично відсутні. Крім того, широкий клас задач побудови та синтезу інтелектуальних мереж, а також їх

діагностики теж формалізується на основі зазначених математичних моделей. Тому є актуальним розроблення методів і архітектури обчислювальних систем, що дозволяють з єдиних позицій розв’язувати зазначені класи задач і з необхідною оперативністю забезпечувати динамічне управління потоками інформації в мережах.

РОЗДІЛ 1. Моделі управління мережами, завдання їх аналізу та синтезу

1.1. Функціональні групи задач управління

Системи управління корпоративними мережами існують не дуже давно. Однією з перших систем такого призначення, що одержала широке поширення, був програмний продукт SunNet Manadger, випущений у 1989 році компанією Sun Soft. SunNet Manadger був орієнтований на управління комунікаційним обладнанням і контроль трафіка мережі. Саме ці функції найчастіше мають на увазі, коли говорять про систему управління мережею. Крім систем управління мережами існують системи управління іншими елементами корпоративної мережі (ОС, СУБД, корпоративними додатками). Застосовуються також системи управління телекомунікаційними мережами: телефонними, а також первинними мережами технологій PDH і SDH.

Незалежно від об'єкта управління, бажано, щоб система управління виконувала ряд функцій, які визначені міжнародними стандартами, узагальнюють досвід застосування систем управління в різних сферах. Існують рекомендації ITU-T X.700 і близький до них стандарт ISO 7498-4, які поділяють завдання системи управління на п'ять функціональних груп:

управління конфігурацією мережі та іменуванням;

обробка помилок;

аналіз продуктивності і надійності;

управління безпекою;

облік роботи мережі.

Розглянемо завдання цих функціональних сфер управління стосовно систем управління мережами.

Управління конфігурацією мережі та іменуванням (*Configuration Management*). Ці завдання полягають у конфігуруванні параметрів як елементів мережі (*Network Element, NE*), так і мережі в цілому. Для елементів мережі, таких як маршрутизатори, мультиплексори і т. п., за допомогою цієї групи завдань визначаються мережеві адреси, ідентифікатори (імена), географічне положення та ін.

Для мережі в цілому управління конфігурацією зазвичай починається з побудови карти мережі, тобто відображення реальних зв'язків між елементами мережі і зміни зв'язків між елементами мережі – утворення нових фізичних або логічних каналів, зміна таблиць комутації та маршрутизації. Управління конфігурацією може виконуватися в автоматичному, напівавтоматичному або ручному режимах. Найчастіше застосовуються напівавтоматичні методи, коли автоматично отриману

карту оператор виправляє вручну. Методи автоматичної побудови топологічної карти, як правило, є фірмовими розробками.

Обробка помилок (*Fault Management*). Ця група завдань включає виявлення, визначення та усунення збоїв і відмов у роботі мережі. На цьому рівні виконується не тільки реєстрація повідомлень про помилки, але і їх фільтрація та аналіз. Виправлення помилок може бути як автоматичним, так і напівавтоматичним. У першому випадку система безпосередньо управляє ресурсами та обладнанням. У напівавтоматичному режимі основні дії з усунення несправності виконують люди, а система управління тільки допомагає в організації цього процесу.

Аналіз продуктивності та надійності (*Performance Management*). Завдання цієї групи пов'язані з оцінкою на основі накопиченої статистичної інформації таких параметрів, як час реакції системи, пропускна здатність реального або віртуального каналів зв'язку між абонентами мережі, інтенсивність трафіка в окремих каналах мережі, імовірність перекручування даних при їх передачі через мережу, а також коефіцієнт готовності мережі. Функції аналізу продуктивності та надійності потрібні як для оперативного управління мережею, так і для планування розвитку мережі.

Управління безпекою (*Security Management*). Завдання цієї групи включають контроль доступу до ресурсів мережі і збереження цілісності даних при їх зберіганні і передачі через мережу. Базовими елементами управління безпекою є процедури аутентифікації користувачів, перевірка прав доступу до ресурсів, розподіл і підтримка ключів шифрування, управління повноваженнями.

Облік роботи мережі (*Accounting Management*). Завдання цієї групи займаються реєстрацією часу використання різних ресурсів мережі: пристроїв, каналів, транспортних служб. Ці завдання мають справу з такими поняттями, як час використання служби і плата за ресурси.

1.2. Багаторівневе подання задач управління

Для побудови інтегрованої системи управління різнорідними елементами мережі природно застосувати багаторівневий ієрархічний підхід. Це, у принципі, стандартний підхід для побудови великої системи будь-якого типу й призначення – від держави до автомобільного заводу. Стосовно систем управління мережами, найбільш опрацьованим і ефективним для створення багаторівневої ієрархічної системи є стандарт *Telecommunication Management Network* (*TMN*), розроблений спільними зусиллями ITU-T, ISO, ANSI і ETSI. Хоча цей стандарт і призначався спочатку для телекомунікаційних мереж, але орієнтація на використання загальних принципів робить його корисним для побудови будь-якої великої інтегрованої розподіленої системи управління. Стандарти TMN

складаються з великої кількості рекомендацій ITU-T, основні принципи моделі TMN описані в рекомендації M.3010.

На кожному рівні ієрархії моделі TMN вирішуються завдання одних і тих самих п'яти функціональних груп, розглянутих вище (тобто управління конфігурацією, продуктивністю, помилками, безпекою, обліком), проте на кожному рівні ці завдання мають свою специфіку. У моделі TMN можна виділити такі основні рівні.

Нижній рівень – *рівень елементів мережі* (*Network Element layer, NE*) – складається з окремих пристроїв мережі: каналів, підсилювачів, кінцевої апаратури, мультиплексорів, комутаторів і т. п. Елементи можуть містити вбудовані засоби для підтримки управління – датчики, інтерфейси. Сучасні технології зазвичай мають вбудовані функції управління, які дозволяють виконувати операції з контролю за станом пристроїв, а також трафіком. Подібні функції вбудовані в технології FDDI, ISDN, Frame Relay, SDH. У цьому випадку пристрій завжди можна охопити системою управління, навіть якщо вона не має спеціального блока управління, тому що протокол технології зобов'язує пристрій підтримувати повні функції управління. Пристрої, які працюють за протоколами, що не мають вбудованих функцій контролю та управління, забезпечуються окремим блоком управління, що підтримує один з двох найбільш поширених протоколів управління – SNMP або CMIP. Ці протоколи належать до прикладного рівня моделі OSI.

Наступний рівень – *рівень управління елементами мережі* (*Network Element Management Layer, NEML*) – являє собою елементарні системи управління. Елементарні системи управління автономно управляють окремими елементами мережі – контролюють канал зв'язку SDH, управляють комутатором або мультиплексором. Рівень управління елементами ізолює верхні шари від деталей і особливостей управління конкретним обладнанням. Цей рівень відповідальний за моделювання поведінки обладнання і функціональних ресурсів нижнього рівня мережі. Атрибути цих моделей дозволяють управляти різними аспектами поведінки керованих ресурсів. Прикладами таких систем можуть служити системи управління Cisco View (Cisco Systems), Optivity (Bay Networks), RAD View (RAD Data Communications) і т. ін.

Вище лежить *рівень управління мережею* (*Network Management Layer*). Цей рівень координує роботу елементарних систем управління, дозволяючи контролювати конфігурацію складових каналів, погоджувати роботу транспортних підмереж.

Наступний рівень – *рівень управління послугами* (*Service Management Layer, SML*) – займається контролем та управлінням транспортними та інформаційними послугами, що надаються кінцевим користувачам мережі. Формування послуги полягає у фіксації в базі даних значень параметрів послуги, наприклад необхідної пропускної здатності, коефіцієнтів готовності і т. п. До функцій цього рівня входить також видача рівнем управління мережею завдання на конфігурування віртуального або

фізичного каналу зв'язку для підтримки послуги. Після формування послуги даний рівень займається контролем за якістю її реалізації.

Рівень бізнес-управління (*Business Management Layer, BML*) займається питаннями довгострокового планування мережі з урахуванням фінансових аспектів діяльності організації, що володіє мережею. Цей рівень є окремим випадком рівня автоматизованої системи управління підприємством (АСУП), ще нижче лежать рівні автоматизованої системи управління технологічними процесами (АСУТП).

1.3. Структура розподілених систем управління

В основі будь-якої системи управління мережею лежить елементарна схема взаємодії агента з менеджером. На основі цієї схеми можуть бути побудовані системи практично будь-якої складності. Схема «менеджер – агент» подана на рис. 1.1.

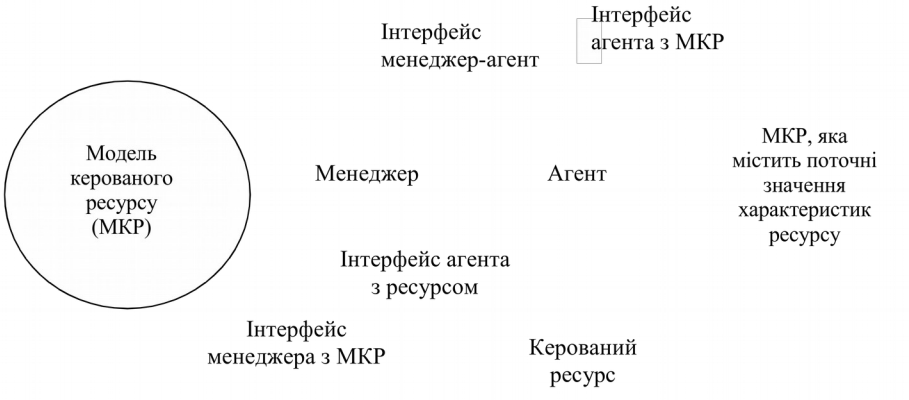


Рис. 1.1. Взаємодія агента, менеджера і керованого ресурсу

Агент є посередником між керованим ресурсом і основною управляючою програмою-менеджером. Щоб один і той самий менеджер міг управляти різними реальними ресурсами, створюється певна модель керованого ресурсу, яка відображає тільки ті характеристики ресурсу, які необхідні для контролю і управління. Менеджер отримує від агента тільки ті дані, які описуються моделлю ресурсу. Агент є деяким екраном, що звільняє менеджера від непотрібної інформації про деталі реалізації ресурсу. Агент поставляє менеджеру оброблену і подану в нормалізованому вигляді інформацію. На основі цієї інформації менеджер приймає рішення з управління та узагальнює дані про стан ресурсу. Менеджер і агент повинні мати у своєму розпорядженні одну і ту саму модель керованого ресурсу, інакше вони не зможуть зрозуміти один одного. Однак у використанні цієї моделі агентом і менеджером є істотна відмінність. Агент наповнює модель керованого ресурсу поточними значеннями характеристик даного ресурсу, і у зв'язку з цим модель агента

називають базою даних управляючої інформації – Management Information Base, MIB. Менеджер використовує модель, щоб знати про те, чим характеризується ресурс, які характеристики він може запросити в агента і яким чином можна управляти.

Менеджер взаємодіє з агентами за стандартним протоколом. Цей протокол дозволяє менеджеру запитувати значення параметрів, що зберігаються в базі MIB, а також передавати агенту управляючу інформацію. Розрізняють управління in-band, тобто по тому самому каналу, по якому передаються дані користувача, і управління out of-band, тобто поза каналом, по якому передаються дані користувача. Спосіб out of-band більш надійний, тому що він надає можливість управляти обладнанням мережі і тоді, коли якісь елементи мережі вийшли з ладу і по основних каналах обладнання недоступне.

Модель «менеджер-агент» лежить в основі таких популярних стандартів управління, як стандарти Internet на основі протоколу SNMP і стандарти управління ISO / OSI на основі протоколу CMIP.

Схема «менеджер-агент» дозволяє будувати досить складні у структурному відношенні розподілені системи управління. Зазвичай розподілена система управління включає велику кількість зв'язків менеджер-агент, які доповнюються робочими станціями операторів мережі, які реалізують доступ до менеджерів (рис. 1.2).

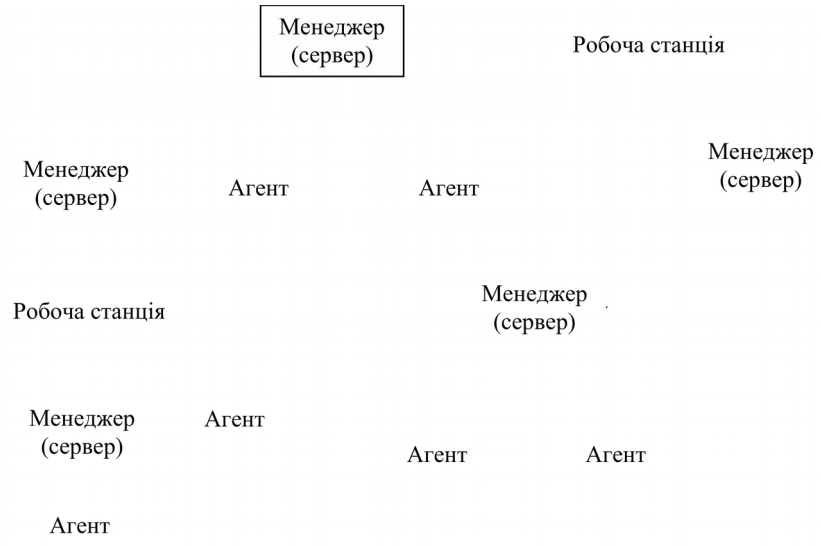


Рис. 1.2. Розподілена система управління на основі кількох менеджерів і робочих станцій

Кожен агент збирає дані і управляє певним елементом мережі. Менеджери, іноді названі серверами системи управління, збирають дані від своїх агентів і зберігають їх у базі даних. Оператори, що працюють за робочими станціями, можуть з'єднатися з будь-якими з менеджерів і за допомогою графічного інтерфейсу проглянути дані про керовану мережу, а

також видати менеджеру деякі директиви з управління мережею та її елементами. Наявність кількох менеджерів дозволяє розподілити між ними навантаження, забезпечуючи масштабованість системи. Більш гнучким є ієрархічна побудова зв'язків між менеджерами. Кожен менеджер нижнього рівня виконує також функції агента для менеджера верхнього рівня. Такий агент працює з набагато більш укрупненою моделлю MIB своєї частини мережі, у якій збирається саме та інформація, яка потрібна менеджеру верхнього рівня для управління мережею в цілому.

Платформний підхід. При побудові систем управління великими локальними і корпоративними мережами зазвичай використовують платформний підхід, коли індивідуальні програми управління розробляються не «з нуля», а використовують служби та примітиви, що надаються спеціально розробленим для цих цілей програмним продуктом (платформою). Прикладами платформ для систем управління є такі відомі продукти, як HP Open View, Sun Net і Manager Sun Soltice, Cabletron Spectrum (IBM) Tivoli TMN10.

Ці платформи створюють загальне операційне середовище для додатків системи управління точно так, як універсальні операційні системи, такі як Unix Windows NT MS Word, Oracle і т. п. Платформа звичайно включає підтримку протоколів взаємодії менеджера з агентами – SNMP і рідше CMIP, набір базових засобів для побудови менеджерів і агентів, а також засоби графічного інтерфейсу.

1.4. Тенденції розвитку телекомунікаційних мереж

Аналіз тенденцій застосування обчислювальних мереж у системах управління реального часу показує, що ефективна їх побудова можлива при повному охопленні автоматизацією всіх її функціональних підсистем, що неможливо на основі повної інтеграції інформаційних ресурсів, тобто створення головної мережі баз даних і баз знань, а також центрів розподілу інформації, доступних керівникам різних рівнів у будь-який час. Створення головної мережі спрямовано на раціональну організацію функціональних процесів і систем доступу, які виконують складні завдання, і на створення сучасної інфраструктури, яка повинна забезпечити виконання завдань супроводу даних окремо від прикладних програм. Впровадження EMBBC (еталонної моделі взаємодії відкритих систем) вимагає розроблення власної ієрархічної системи стандартних протоколів для відкритих систем. Інтеграція ведеться на базі єдиного апаратного та програмного забезпечення всіх систем. При цьому можна виділити три підсистеми. Перша підсистема забезпечує органи управління первинною інформацією, що надходить від джерел, що діють на всьому просторі роботи системи. Вона узагальнює ці дані для того, щоб надати керівникам усіх рівнів єдину інформаційну картину про стан всіх об'єктів системи. Друга підсистема призначена для збору, обробки, об'єднання та аналізу первинних даних, а

також для їх збереження і передачі по запитах споживачів. Третя підсистема являє собою засоби, що забезпечують прийняття рішень з управління та розв’язання необхідних прикладних задач. Такий підхід, на думку фахівців, дає три важливі переваги: оперативне досягнення необхідної мети функціонування системи, відповідність критерію «вартість/ефективність», зниження ризику непродуктивних витрат. Таким чином, кінцева мета створюваної глобальної системи управління – це збирання, обробка, аналіз і розподіл інформації, що забезпечує високий ступень надійності і швидкості, які дозволяють управляти процесами, що швидко протікають.

Для вирішення завдання інтеграції систем управління з різною спеціалізацією покладено принцип створення єдиного операційного середовища, інформаційної інфраструктури всієї системи в цілому. Ядро середовища утворюють операційні системи ЕОМ, об'єднані в єдину операційну систему мережі, структури даних і програмних додатків, формати друкованих документів і вид інформації, що відображається. Це дозволяє виконувати завдання, призначені для звичайних персональних комп'ютерів, на ЕОМ, встановлених і на рухомих об'єктах.

До глобальної системи управління висуваються такі основні вимоги: однозначне подання оперативної інформації для всіх об'єктів управління; швидке перетворення всієї інформації, що надходить у стандартній формі, забезпечення інформацією операторів за запитами та організація інтелектуального діалогу з користувачем; подання даних у зручних для сприйняття формах (голосова інформація, відеозображення та ін.); забезпечення всіх користувачів єдиним стандартизованим робочим місцем оператора; наявність розвинених інтелектуальних засобів аналізу та обробки інформації; використання модульного принципу побудови комплексів апаратури для спрощення їх модернізації і зниження вартості.

Всю інформацію, що циркулює в об'єднаній інформаційній системі, можна розділити на глобальну і оперативну. Мережі обміну глобальною інформацією отримують її від будь-яких джерел. Це можуть бути бази даних різних центрів управління. Зібрана інформація потім передається на комплекси обробки даних керівництва. Тут вона порівнюється з інформацією, що раніше надійшла, доповнюється даними, отриманими в результаті планування та прийняття рішення на управління, і передається далі в мережі обміну оперативною інформацією. У них вона може поділятися на категорії залежно від цілей функціонування підсистем і надходити на центри управління, де здійснюється планування операцій з виробленням оптимального варіанта їх проведення, внесення в нього оперативних коректив з урахуванням мінливої обстановки. Уся оновлена інформація про обстановку надається іншим користувачам. Тут же проводиться попередня обробка даних.

Мережі обміну глобальною інформацією являють собою віртуальні мережі обміну даними, що зв'язують різні центри і пункти управління. На

сьогодні визначена концепція спеціалізації мереж виходячи з їх функціонального призначення: передачі зображень; передачі управляючої інформації (у тому числі забезпечення проведення відео- і телеконференцій, голосовий обмін, факсимільний і телетайпний зв'язок); управління базами даних; науково-дослідницькою інформацією; інформаційний обмін текстовими повідомленнями і т. ін.

Кількість віртуальних мереж повинна мати можливість нарощуватися за умови, що вони будуть задовольняти загальні вимоги, використовувати єдиний формат повідомлень і протокол обміну.

Використання мереж у системах управління реального часу дозволяє істотно підвищити гнучкість системи управління, поліпшити якість прийнятих рішень. При цьому функціонування обчислювальних мереж має свої особливості, пов'язані з необхідністю більш швидкого і гнучкого розподілу інформаційних потоків. Це вимагає розроблення нових принципів відмовостійкого функціонування обчислювальних мереж, які передбачають максимально можливу оперативність управління загальносистемними ресурсами і потоками інформації в них.

1.5. Особливості управління в телекомунікаційних мережах, які використовуються в системах управління в реальному часі

Особливістю управління в телекомунікаційних мережах, що працюють у системах управління в реальному часі, є те, що вони повинні володіти високою продуктивністю, відрізнятися простотою і високою надійністю в складних умовах експлуатації. Управління в мережі здійснюється за допомогою системи протоколів, що є частиною операційної системи мережі (ОСМ). Умовно можна виділити три основні функції управління: управління мережею передачі даних; управління зв'язком в обчислювальній мережі; управління взаємодією процесів користувачів в обчислювальній мережі.

При управлінні потоками інформації в мережах під процесом розуміють [1, 2, 18, 28-31] програми користувачів, що визначають обчислювальні ресурси і здійснюють обробку даних в обчислювальних системах і термінальних пристроях мережі. Протоколом називають систему правил обміну повідомленнями між процесами, що забезпечує логічну та часову синхронізацію при обміні повідомленнями, і угоди за форматами переданих повідомлень. Управління мережею передачі даних включає в себе збір та обробку інформації про стан мережі, вироблення рішень з управління мережею, управління потоками інформації в мережі, управління навантаженням і структурою мережі. Під управлінням зв'язком в обчислювальній мережі [10, 17, 36] розуміють управління прийманням і передачею повідомлень, управління комутацією потоків пакетів, управління інформаційним каналом і якістю передачі інформації. До основних функцій управління взаємодією процесів користувачів віднесемо такі: управління

діалоговими режимами в мережі, управління віддаленою обробкою задач, обмінами масивами інформації, управління розв'язанням задач і використанням обчислювальних ресурсів мережі. Потреба доставки повідомлень з одного пункту мережі *i* в інший *j* задають тяжінням-об'ємом повідомлень, який необхідно доставити з *i* в *j* за заданий проміжок часу. Тяжіння в мережі визначаються взаємодією процесів у мережі. Інтенсивність тяжіння в мережі постійно змінюється. Цей процес у момент часу *t* можна задати у вигляді матриці вимог

$$M(t) = | \varphi_{ij}(t) |,$$

де φ_{ij} – інтенсивність вимог заявок на передачу повідомлень з вузла *i* у вузол *j* у момент часу *t*.

Вимоги можуть виражатися у вигляді числа каналів із заданою пропускною здатністю, середнього числа заявок на передачу інформації з певною швидкістю. У загальному випадку процес у потребі зв'язку між абонентами мережі може бути описаний у вигляді послідовності змін матриць вимог

$$M(t_1) \Rightarrow M(t_2) \Rightarrow \ldots\ldots \quad \Rightarrow M(t_i) \ldots\ldots \Rightarrow M(t_K). \qquad (1.1)$$

Отже, завдання управління потоками інформації в мережі можна розглядати як задачу забезпечення реалізації вимог (1.1) у будь-який момент часу з заданою якістю або ефективністю обслуговування.

Для розв'язання цієї задачі відомо два підходи. Перший підхід полягає у створенні мережі за так званою максимальною матрицею вимог

$$M_{\max} = | \varphi_{ij}^{\max} |,$$

кожен елемент якої дорівнює максимальному з однойменних елементів всіх матриць вимог процесу (1.1), тобто

$$\varphi_{ij}^{\max} = [\varphi_{ij}(t_1), \ldots, \varphi_{ij}(t_i), \ldots, \varphi_{ij}(t_k)].$$

У цьому випадку організація управління мережею передбачає, що обрано структуру мережі, методи комутації та план розподілу інформації. Однак це призводить до збитковості мережі і надмірного збільшення її вартості. Причому надмірність тим більше, чим більше розходження в значеннях однойменних елементів матриць вимог процесу (1.1). Тому було запропоновано більш перспективний шлях розв'язання цієї задачі, при якому мережа створюється за деякою фіктивною матрицею вимог

$$M_{ф} = | \varphi_{ij}^{ф} |. \qquad (1.2)$$

Матриця (1.2) складається так, що побудована за нею мережа забезпечує задану ефективність обслуговування для будь-якої матриці вимог процесу (1.1) за умови перерозподілу інформації від $M(t_i)$ до $M(t_j)$. Для адаптивного перерозподілу інформації, з урахуванням ситуації, що змінюється в мережі, створюється система управління потоками інформації, названа динамічною системою управління потоками інформації. Управління в мережі повинно забезпечувати найбільш ефективний режим використання ресурсів мережі при змінних умовах її функціонування. У мережах спеціального призначення можна виділити два взаємопов'язані процеси: управління розв'язанням задач у мережі в інтересах системи управління і управління потоками інформації та взаємодією процесів користувачів. Причому процес розв'язання задач у мережі передбачає централізоване управління, а управління потоками інформації повинно носити децентралізований характер для забезпечення відмовостійкого функціонування мережі.

Як будь-яка система управління, система динамічного управління потоками інформації об'єднує різні об'єкти, між якими передається інформація про стан і команди управління.

Об'єкти управління – це пристрої або програми, що забезпечують на вузлах комутації реалізацію процесу розподілу потоків згідно з адресами, матрицями маршрутів на основі процедури обслуговування. Крім того, на об'єктах управління здійснюється контроль стану елементів мережі.

Управляючі об'єкти – це пристрої або програми, що здійснюють збір інформації про характеристики і стан елементів мережі та формування матриць маршрутів. Інформація стану включає інформацію двох видів: власне інформацію стану і статистичну інформацію про характеристики елементів мережі і потоків повідомлень. Ініціювання контуру управління може бути програмним, ситуаційним та інтервальним. У випадку програмного управління певний набір матриць маршрутів змінюється за певним розкладом, що складається на основі апріорних відомостей про розподіл потоків у часі. При ситуаційному управлінні рішення про перебудову матриць маршрутів приймається на основі поточного контролю ваг. Контур управління ініціюється при переході значень ваг через певні порогові величини. Інтервальне управління передбачає реалізацію контуру управління через фіксовані інтервали часу. Важливим питанням при забезпеченні управління потоками є передача службової інформації. Найбільш широко застосовуються для управління потоками інформації такі засоби:

1. Збір інформації. При цьому на мережі утворюється спеціальна підмережа, що складається зі спеціально виділених каналів, призначених для передачі тільки службової інформації.

2. Використання службових повідомлень.

3. Використання інформаційних повідомлень (у складі службової частини кожного інформаційного повідомлення передбачається місце для службової інформації).

4. Використання "зонд-повідомлень" (формування матриць маршрутів реалізується шляхом натурного експерименту).

Для обмеження навантажень у мережах використовується міжкінцеве і глобальне управління [19, 22, 27].

З урахуванням досліджень, проведених у роботах [12, 17, 21, 27], можна виділити такі основні завдання, які вирішуються динамічною системою управління потоками інформації та взаємодією процесів у телекомунікаційній мережі:

1. Управління розв'язанням задач і використанням обчислювальних ресурсів мережі в умовах її деградації.

2. Управління маршрутизацією повідомлень із забезпеченням заданої прихованості передачі інформації в мережі.

3. Оцінка пропускної здатності в умовах деградації мережі.

4. Забезпечення адаптивного до змінних потоків задач управління у вузлах мережі.

5. Оцінка стану мережі та відновлення мережі у випадках відмов її функціональних елементів.

6. Адаптивне відображення логічної структури бази даних мережі на її фізичну структуру в умовах деградації мережі.

7. Планування показу рухомих центрів управління і комутації мережі за наявності рухомих пунктів управління.

Як показано в роботах [20, 25, 28, 30], розв'язання даного комплексу задач у мережах спеціального призначення повинно здійснюватися в масштабі реального часу.

1.6. Показники ефективності функціонування телекомунікаційних мереж

Показник ефективності $W(u)$ операції є мірою відповідності реального результату необхідному ($u \in \Omega^v$ - обрана стратегія з безлічі припустимих) [63]. У відмовостійкої НД під операцією розуміється процес перерозподілу ІРЗ від ПМ НД між р-ПМ. Основною вимогою при виборі показника ефективності є відповідність показника мети операції, яка відображається потрібним результатом Y^{mp}. Тому для опису відповідності реального результату Y операції необхідному формально вводиться числова функція на безлічі результатів операції

$$\rho = \rho\,(Y(u),\, Y^{mp}),\, u \in \Omega^v, \qquad (1.3)$$

яку називають функцією відповідності.

У разі опису детермінованих процесів [63] функція відповідності служить показником ефективності, тобто

$$W(u) = (Y(u),\ Y^{mp}). \qquad (1.4)$$

Результат операції $Y(v)$ ставлять у залежність від основних результуючих факторів – корисного ефекту E, витрачених ресурсів C і часу T [63]. У загальному випадку функція $Y(u)$ може бути подана як степенева функція від результуючих факторів

$$Y(u) = \alpha_0 E^{\alpha_1} C^{\alpha_2} T^{\alpha_3}, \qquad (1.5)$$

де α_0, α_1, α_2, α_3 – параметри функції результату.

Якщо покласти $\alpha_0 = \alpha_1 = 1$ і $\alpha_2 = \alpha_3 = 0$, то результат операції описується лише результуючим фактором E, тобто $Y(v) = E$, а на решту результуючих факторів будуть накладені обмеження.

При побудові відмовостійкої мережі результуючий фактор E характеризується коефіцієнтом збереження ефективності $k_{се}$, що дорівнює відношенню значення показника ефективності використання мережі за призначенням за певну тривалість часу експлуатації до номінального значення цього показника, вирахуваного за умови, що відмови в мережі протягом того самого періоду не виникають [23, 32, 37]. Коефіцієнт збереження ефективності характеризує ступінь впливу відмов на ефективність застосування мережі за призначенням, тобто

$$k_{сэ} = \frac{П_{отк}}{П_{ном}}, \qquad (1.6)$$

де $П_{отк}$ – значення показника ефективності НД при відмовах;

$П_{ном}$ – номінальне значення показника ефективності НД.

Оскільки системи управління реального часу є складними системами, то для оцінки їх можливостей використовуються різні показники ефективності, наприклад оперативність розв’язання ІРЗ, пропускна здатність системи, мобільність, стійкість, ємність з обробки інформації і т. д., які можна розділити на дві групи [23, 27]:

1) $\varphi_к$ – показники, що оцінюють функціональні можливості системи. Одним з таких показників може служити функціональна потужність системи E_c, обумовлена безліччю задач, які здатна розв’язувати мережа в даний момент або протягом певного інтервалу часу. Цей показник може залежати не тільки від кількості задач, але й від їх складності, важливості і від інших характеристик, не пов'язаних з часом розв’язання ІРЗ;

2) θ_m – показники, що оцінюють час або швидкість виконання системою її функцій. Цю групу складають показники продуктивності і пропускної здатності системи, до яких відносять і такі показники: середній час розв’язання в даній мережі ІРЗ, середній час перебування в мережі заявки на розв’язання будь-якої з допустимих для даної системи

управління задачі, оперативність розв'язання задач управління в мережі, оперативність розв'язання ІРЗ в мережі і подібні до них показники. У відмовостійких мережах, що мають здатність до перерозподілу ІРЗ при відмовах ПМ, можлива деградація двох видів [23, 27]:

1. Функціональна деградація, яка виражається в зниженні значень показників групи $\{\varphi_k\}$ і призводить до функціональної відмови (φ - відмови), що відповідає виникненню події $A_\phi = \{ \varphi' < \varphi^* \}$, де φ', φ^* - поточне і гранично допустиме значення деякого показника j, прийнятого для оцінки мережі.

2. Деградація за продуктивністю або оперативністю, тобто зниження показників групи (θ_m), що призводить до відмови через продуктивність або оперативність (θ - відмови), під яким розуміється настання події $A_0 = \{\theta' < \theta^*\}$, де θ', θ^* - поточне і гранично допустиме значення деякого показника q, що використовується для оцінки мережі.

У роботі [27] розглядаються два типи станів системи: $æ$ - стан, що визначається парою значень прийнятих показників φ і θ; s - стан, обумовлений як $s_\nu = \sigma_{1,\dots,}\sigma_n$, де $\sigma_i \in \{0,1\}$, $\sigma_i = 1$ для р-ПМ і $\sigma_i = 0$ для р-ПМ. При цьому $X^0 = \{\varphi^0, \theta^0\}$ - початковий æ-стан, φ^0, θ^0 - номінальні значення показників φ і θ; $s^0 = 00...0$ - початковий s - стан.

Номінальні значення φ^0, θ^0 і гранично допустимі значення φ^*, θ^* прийнятих показників якості функціонування визначають область працездатності системи (P - область) як підмножину λr = (æμ) таких æ - станів, для яких $\varphi^0 \geq \varphi_\mu \geq \varphi^*$ та $\theta^0 \geq \theta_\mu \geq \theta^*$ (рис. 1.3).

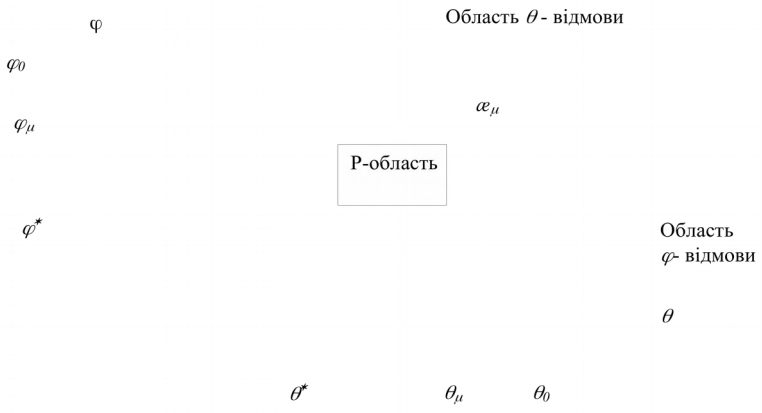


Рис. 1.3. Область працездатності системи

У кожній системі управління повинно бути реалізовано те чи інше відображення φ: $X_s \rightarrow X_æ$ безлічі $X_s = \{s_\nu\}$ всіх s-станів у безліч $X_æ = \{x_\mu\}$ всіх æ-станів, при цьому має бути забезпечено відображення у P - область λ_r (де $\lambda_r \in X_æ$) такої підмножини $S \subset X_s$, щоб виконувалися поставлені вимоги до відмовостійкості мережі. Реалізацію цього відображення можна виконати

шляхом належного РЗ, що розв'язуються системою, між р-ПМ для кожного зі станів $s_v \in X_s$.

У цьому випадку результат операції визначається корисним ефектом E_v, оцінюваним сумарною ваговою характеристикою безлічі ІРЗ в даному S_n-стані, що отримала назву функціональної потужності системи [63-70]

$$E_v(\vartheta_\mu^i) = \sum_{i=1}^{n_v} \sum_{\vartheta_\mu^i \in \Omega_i^v} \beta_{\mu i}, \ \mu = \overline{1, m_i}, \tag{1.7}$$

де n_v - загальна кількість ПМ в мережі в стані S_n;

$\beta_{\mu i}$ - вага μ-ї задачі i-го ПМ, що характеризує його важливість (пріоритет);

m_i-загальна кількість задач, що знаходяться на розв'язанні в i-му ПМ;

$\vartheta_\mu^i \in \Omega_i^v$ - безліч задач, які здатний розв'язувати i-й ПМ при S_v.

Таким чином, основним показником $W(u)$ ефективності мережі, що найбільш повно відображає цілі операції, є скаляр (1.7), який називається показником середнього результату [63], тобто

$$W(u) = M[Y(u)] = E_v]. \tag{1.8}$$

Тоді відповідно до формули (1.6) при виникненні φ-відмови в мережі коефіцієнт збереження ефективності буде мати вигляд

$$K_{\text{се}} = \frac{E_v + E_в(\vartheta)}{E_0}; \ E_в(\vartheta) = \left| E_v^*(\vartheta) - E_v \right|, \tag{1.9}$$

де $E_в$ – корисний ефект операції;

E_v - значення після проведення операції РЗ в системі управління;

E_0 – номінальне значення функціональної потужності безлічі ІРЗ в мережі.

Показник E_v належить до групи $\{\varphi_k\}$ - показників, до якої також належить і інший результуючий фактор формули (1.5) - витрати на використання будь-якого (нетимчасового) ресурсу ПМ при S_v. Таким, наприклад, може бути обсяг оперативної пам'яті займаної ІРЗ μ в ПМ M_i в R-му р-ПМ ($R \in$ □$_v$ – безлічі працездатних ПМ при n), який задається матрицею

$$\left| C_{\mu i}^R \right| \left(\mu = \overline{1, m_i}; \quad i = \overline{1, n_v}; \quad R \in D_v^R \right).$$

Звичайно на кожну операцію накладають обмеження значення тимчасових результуючих факторів формули (1.5), що належать до групи $\{\theta_m\}$-показників. Таким, наприклад, показником може бути середній час

обслуговування (перебування) ІРЗ i-го ПМ в R-му в р-ПМ, яке задається матрицею

$$\left| \Delta T_{\mu i}^{R} \right| \left(\mu = \overline{1, m_i}; \quad i = \overline{1, n_v}; \quad R \in D_v^R \right).$$

Для оцінки часу проведення операції важливу роль відіграють часові показники мережі. Таким показником може виступати час пересилання (затримки) $\left| t_{\mu i}^{R} \right|$ повідомлень ІРЗ μ i-го ПМ при передачі в R-й р-ПМ.

Для вибору критерію ефективності функціонування відмовостійкої мережі використовуємо концепцію оптимізації [63], яка вважає раціональними ті стратегії $u \in \Omega^v$, які забезпечують максимальний ефект операції, тобто

$$W(u^*) = \max_{u \in \Omega^v} W(u). \tag{1.10}$$

Оптимальна стратегія може бути не єдиною, тобто розв'язання задачі (1.10) може дати безліч рівноцінних оптимальних стратегій $u^* \in \Omega^v$.

Залежно від виду функції відповідності ρ в рамках концепції оптимізації критерієм оптимальності операції буде критерій найбільшого результату [63]. Оптимальну стратегію вибирають з умови (1.10), тобто

$$u^* : \max_{u \in \Omega^v} E_v(u). \tag{1.11}$$

Таким чином, у кожний момент v, що задається циклом управління, необхідно здійснювати аналіз вектора s_v - стану та при виникненні φ - відмови здійснювати пошук оптимальної стратегії u^* за формулою (1.11) операції з метою досягнення її корисного максимального ефекту $E_в$ і підвищення коефіцієнта збереження ефективності за формулою (1.9).

1.7. Облік часових характеристик при динамічному управлінні в мережах

Ефективність функціонування динамічної системи управління та мережі в цілому характеризується такими показниками якості: математичне очікування $M[\varphi_c]$ сумарного обсягу потоку інформації, пропущеного в мережі за період оновлення інформації в мережі $T_{обн}$. Як відомо [14, 22, 28-31], цикл оновлення інформації на мережах задається виходячи з функціонального призначення мережі, і в сучасних глобальних інформаційно-обчислювальних мережах становить від кількох секунд до кількох хвилин.

Цикл оновлення інформації про стан мережі (*цикл управління мережею*) – інтервал часу, з періодичністю якого управляючий файл-сервер

робить опитування стану вузлів мережі і вузлів комутації з метою отримання детальної інформації (стан вузла, працездатність каналу з'єднання, наявність вільних і зайнятих ресурсів, тип і характеристики виникаючих збоїв і відмов). Цикл оновлення інформації в мережі такий:

$$T_{обн} = t_{відпр} + t_{розп} + t_{дек} + t_{ан} + t_{Р} + t_{код} + t_{уп} + t_{ср} + t_{ане}, \quad (1.12)$$

де $t_{відпр}$ – час відправлення сигналу опитування по вузлах та отримання даних про стан мережі;

$t_{розп}$ – час розпакування одержуваних пакетів даних;

$t_{дек}$ – час декодування отриманих даних (за необхідності);

$t_{ан}$ – час, що витрачається на попередній аналіз отриманих даних, наприклад сортування або отримання (вхідних-вихідних) даних, для вбудованих алгоритмів і т. п.;

$t_{Р}$ – час розв'язання задач динамічного управління в мережі;

$t_{код}$, $t_{уп}$ – зворотні процедури кодування та упакування переданих даних;

$t_{ср}$ – час передачі сигналу розсилки вузлів, даних і отримання відповідного луна-сигналу;

$t_{ане}$ – час аналізу луна-сигналу з метою перевірки на наявність якісного приймання.

Залежно від спеціалізації мереж умови, які повинен задовольняти цикл оновлення інформації в мережі, лежать у межах від 0,2 до 30 с [28-31].

Оперативність розв'язання задач динамічного управління в мережі за час, що не перевищує допустимий $T_д$, оцінюють кількісно ймовірністю розв'язання комплексу задач. $P(T)$ за час T не перевищує $T_Д$ [202]

$$P(T) = 1 - e^{\frac{T_д}{T}}. \quad (1.13)$$

У мережах, призначених для оперативного розв'язання задач, у якості узагальненого показника оперативності розв'язання задач управління використовують [202] імовірність розв'язання задач управління в системі управління протягом інтервалу часу $\Delta(t) = T_{кр} - T_{цу}$ ($T_{кр}$ - критичний час, після якого мета управління не може бути досягнута)

$$P[\Delta(t)] = K_г e^{-\frac{T_{кр} - T_{цу}}{T_0}}, \quad (1.14)$$

де $K_г$ – коефіцієнт готовності системи;

T_0 – середній час напрацювання системи на одну відмову;

$T_{цу}$ – час циклу управління в системі.

1.8. Математичні моделі задач управління плануванням і побудовою телекомунікаційних мереж

Розгляд математичних моделей почнемо з найбільш важливої задачі розподілу задач у мережі.

1.8.1. Розподіл задач у мережі

Вихідними даними для розв’язання цієї задачі є виконувані системою функції, які можуть бути формалізовані у вигляді безлічі розв’язуваних задач. Для кожної з задач можуть бути задані можливі варіанти її розв’язання, і кожна задача може складатися з декількох етапів. Зв'язок між задачами та етапами задаються у вигляді графа, ребра якого характеризують співвідношення прямування, що існують між розв'язуваними задачами і їх етапами, і відповідають напрямам інформаційних потоків. Крім того, задається безліч вузлів мережі і зв'язків між ними і види технічних засобів, які використовуються. Потрібно таким чином розподілити задачі по вузлах і рівнях системи, при яких максимізується ефект W від розв’язання.

Часові обмеження для різних задач вимагають аналізу роботи різних вузлів мережі. Як показано в роботах [2, 22, 37], ця задача належить до класу задач нелінійного булевого програмування, для яких немає ефективних методів розв’язання, тому на практиці розв’язують окремі задачі оптимізації розподілу. При цьому в якості критеріїв оптимізації використовують такі критерії:

1. Мінімізація витрат на реалізацію задач у системі

$$\min \sum_{i=1}^{I} \sum_{j=1}^{J} W_{ij} X_{ij}, \tag{1.15}$$

де $i = \overline{1, I}$ – безліч задач, функцій, що реалізуються;

$j = \overline{1, J}$ – безліч обслуговуючих вузлів системи;

W_{ij} - витрати на реалізацію i-ї задачі в j-му вузлі;

$X_{ij} = \begin{cases} 1, \text{якщо задача i розв'язується у j - му вузлі;} \\ 0, \text{в іншому випадку.} \end{cases}$

2. Мінімізація загального часу розв'язання всіх задач

$$\min \sum_{i=1}^{I} \sum_{j=1}^{J} t_{ij} x_{ij}, \tag{1.16}$$

де t_{ij} – час розв’язання i-ї задачі в j-му вузлі.

3. Мінімізація максимального часу розв'язання задач у системі

$$\min \left\{ \max_{j} \sum_{i=1}^{I} t_{ij} x_{ij} \right\}. \tag{1.17}$$

Можлива оптимізація за більш складними критеріями: забезпечення необхідного часу готовності системи і т. д. Обмеження в окремих задачах оптимізації використовуються такі:

1. На зв'язок між задачами, тобто заданий граф G_F.
2. На зв'язок між вузлами, тобто заданий граф G_M.
3. На загальні витрати на реалізацію задач у системі

$$\sum_{i=1}^{I} \sum_{j=1}^{J} W_{ij} X_{ij} \le W_{доп}. \tag{1.18}$$

4. На витрати на реалізацію задач у вузлах

$$\sum_{i=1}^{I} W_{ij} X_{ij} \le W_{jдоп}, \quad j = \overline{1, J}. \tag{1.19}$$

5. На завантаження кожного вузла

$$\sum_{i=1}^{I} \lambda_i t_{ij} x_{ij} \le \rho_j, \quad j = \overline{1, J}, \tag{1.20}$$

де λ_i – інтенсивність надходження i-ї задачі на розв’язання.

6. На загальний час розв’язання задачі

$$\sum_{i=1}^{I} \sum_{j=1}^{J} t_{ij} x_{ij} \le T. \tag{1.21}$$

7. На час розв’язання окремих задач

$$\sum_{j=1}^{J} t_{ij} x_{ij} \le \tau_i, \quad i = \overline{1, I}. \tag{1.22}$$

Розглянемо основні типи окремих задач оптимізації.

Перша задача оптимізації процесу розподілу

Необхідно розподілити i задач $i = \overline{1, I}$ між j вузлами $j = \overline{1, J}$, щоб забезпечити мінімум загальних витрат часу (формула (1.21)) або мінімум загального часу розв'язання (формула (1.22)) при виконанні обмежень на завантаження кожного з вузлів формули (1.20) або витрати в кожному j вузлі (формула (1.19)). Математична модель цієї задачі може бути записана так. Знайти

$$\min \sum_{i=1}^{I} \alpha_{ij} x_{ij} \tag{1.23}$$

при обмеженнях

$$\sum_{i=1}^{I} \beta_{ij} X_{ij} \le b_j \quad j = \overline{1, J}; \tag{1.24}$$

$$\sum_{j=1}^{J} X_{ij} = 1, \quad i = \overline{1, I}, \quad X_{ij} \in \{0,1\}. \tag{1.25}$$

У співвідношеннях (1.23) – (1.25) a_{ij} – витрати (або час розв'язання) i-ї задачі в j-му вузлі; b_j-допустимі витрати (завантаження) в j-му вузлі;

$$X_{ij} = \begin{cases} 1, \text{ якщо задача } i \text{ розв'язується у } j \text{ - му вузлі;} \\ 0, \text{ в іншому випадку.} \end{cases}$$

Обмеження (1.25) означає, що кожна задача розв'язується тільки в одному вузлі.

Друга задача оптимізації процесу розподілу задач

Необхідно так розподілити i задач $i = \overline{1, I}$ між j вузлами, $j = \overline{1, J}$, щоб забезпечити мінімум часу загальних витрат (формула (1.21)) або мінімум загального часу розв'язання (формула (1.22)) при виконанні обмежень на загальний час розв'язання (формула (1.21)) або загальні витрати (формула (1.24)). Математична модель цієї задачі може бути записана так. Знайти

$$\min \sum_{i=1}^{I} \sum_{j=1}^{J} \alpha_{ij} x_{ij} \tag{1.26}$$

при обмеженнях

$$\min \sum_{i=1}^{I} \sum_{j=1}^{J} \beta_{ij} x_{ij} \le B, \tag{1.27}$$

$$\sum_{j=1}^{J} X_{ij} = 1, \quad i = \overline{1, I}, \quad X_{ij} \in \{0,1\}. \tag{1.28}$$

У співвідношеннях (1.26) – (1.28) a_{ij} – витрати (або час розв'язання) i-ї задачі в j-му вузлі; b_{ij} – час розв'язання (або витрати) і-ї задачі в j-му вузлі; B – загальний час розв'язання (або витрати) усіх задач.

Третя окрема задача оптимізації процесу розподілу задач

Необхідно так розподілити i задач $i = \overline{1, I}$ між j вузлами, $j = \overline{1, J}$, щоб забезпечити мінімум часу загальних витрат (формула (1.21)) або мінімум загального часу розв'язання (формула (1.22)) при виконанні обмежень на загальний час розв'язання (формула (1.21)) і завантаження вузлів (формула (1.20)), або на загальні витрати (формула (1.24)) і завантаження вузлів (формула (1.20)). Математична модель цієї задачі має такий вигляд. Знайти

$$\min \sum_{i=1}^{I} \sum_{j=1}^{J} \alpha_{ij} x_{ij} \tag{1.29}$$

при обмеженнях

$$\min \sum_{i=1}^{I} \sum_{j=1}^{J} \beta_{ij} x_{ij} \le B, \tag{1.30}$$

$$\sum_{i=1}^{I} \sum_{j=1}^{J} C_{ij} X_{ij} \le C_j, \quad j = \overline{1, J}, \tag{1.31}$$

$$\sum_{j=1}^{J} X_{ij} = 1, \quad i = \overline{1, I}, \quad X_{ij} \in \{0,1\}. \tag{1.32}$$

Розглянуті задачі є основними [22, 61, 62] при оптимізації процесу розподілу задач у телекомунікаційних мережах. Особливості оптимізації цього процесу в умовах деградації мережі будуть розглянуті в розд. 5.

1.8.2. Оптимальні маршрути в телекомунікаційних мережах

Поняття оптимального маршруту є умовним і визначається обраним критерієм. У свою чергу критерій оптимального шляху залежить від показника, що використовується як вага елементів мережі. Все різноманіття можливих критеріїв утворює три класи [22, 27, 34-35].

До *першого класу* відносять критерії, засновані на адитивних показниках, таких як довжини ліній зв'язку, вартість передачі повідомлень, параметри затримки. При цьому критерій оптимального маршруту має вигляд

$$\min_{S_k \in \Omega} \sum_{(i,j) \in S_k} q_{ij}, \tag{1.33}$$

де S_k – k-й маршрут між деякою парою вузлів;

Ω – множина всіх таких маршрутів.

Другий клас утворюють критерії, засновані на мультиплікативних показниках, зазвичай імовірнісного змісту. Сюди належать коефіцієнт справної дії; імовірність безпомилкової передачі; імовірність своєчасної доставки. Для цього класу узагальнений запис критерію такий:

$$\max_{S_k \in \Omega} \prod_{(i,j) \in S_k} q_{ij}. \tag{1.34}$$

До *третього класу* належить критерій Максміна, фізичним параметром для якого є, наприклад, пропускна здатність елементів мережі. Даний критерій має вигляд

$$\max_{S_k \in \Omega} \min_{(i,j) \in S_k} q_{ij}. \tag{1.35}$$

При цьому оптимальним буде маршрут, що має максимальну пропускну здатність. Критерій (1.35) не приводиться до загального вигляду формули (1.33), що необхідно враховувати при його використанні в задачах маршрутизації. Формальною моделлю задачі маршрутизації є задача визначення найкоротшого шляху в мережі, заданої графом $G(V,E)$, ребрам якого приписані ваги, що задаються матрицею довжин $L = [l_{ij}]$.

Задача про найкоротший шлях полягає в знаходженні найкоротшого шляху від заданої початкової вершини $s \in V$ до заданої кінцевої вершини $t \in V$ за умови, що такий шлях існує. При роботі мережі в умовах її деградації виникає необхідність у визначенні найбільш надійних шляхів, тобто замість матриці довжин задається матриця $[p_{ij}]$, де p_{ij} – імовірність того, що лінія зв'язку, відповідна ребру (i, j), знаходиться в справному стані. Надійність шляху від s до t, складеного з ребер, взятих з безлічі K, визначиться співвідношенням

$$P(K) = \prod_{(i,j) \in K} p_{ij}. \tag{1.36}$$

Ця задача зводиться до задачі про найкоротший шлях від s до t, взявши в якості ваги ребра (i, j) величину $c_{ij} = -\log(p_{ij})$.

Узявши логарифм, отримаємо

$$\log P(K) = \sum_{(i,j) \in K} \log(p_{ij}) = -\sum_{(i,j) \in K} c_{ij}. \tag{1.37}$$

Звідси видно, що найкоротший шлях від s до t з матрицею ваг $[c_{ij}]$ буде в той же час і найбільш надійним шляхом з матрицею $[p_{ij}]$, а надійність цього шляху дорівнює антилогарифму його довжини. Решта моделей комплексу задач (1÷7) п. 1.2, як показано в роботі [161], теж належить до класу задач комбінаторної оптимізації і більш докладно буде розглянута у розд. 5.

1.8.3. Математичні моделі задач, розв'язуваних при побудові інтелектуальних телекомунікаційних мереж

Практикою побудови інтелектуальних систем для автоматизації процесу прийняття рішень встановлено [38, 118, 303-307], що процес вироблення рішень складається з таких складових компонент: з'ясування поставленої задачі, оцінка обстановки, вироблення задуму розв'язання, формалізація процесу дій, оптимізація математичної моделі, аналіз результатів, формулювання розв'язання, постановка задачі перед виконавцями. Математична модель операцій встановлює відповідність між значеннями керованих і некерованих змінних.

У загальному вигляді математична модель може бути подана у вигляді

$$W = f(x_1, \ldots, x_m; y_1, \ldots, y_n),\ Q_k(x, \ldots, x; y_1, \ldots, y_n) = 0,\ k = \overline{1, p},$$

де W – критерій ефективності;

$x_i,\ i = \overline{1, m}$ – керовані змінні;

$y_j, j = \overline{1, n}$ – некеровані змінні або випадкові впливи;

$Q_k = \overline{1, p}$ – функції, що виражають обмеження.

Введення обмежень істотно збільшує складність моделі. У цих випадках, навіть якщо критерій ефективності представляє диференційовану функцію, максимум не може бути визначений за допомогою класичних методів математичного аналізу за умови, що максимум не лежить всередині області з певною сукупністю обмежень. У тих випадках, коли критерій ефективності взагалі не можна диференціювати або кількість змінних велика, знайти оптимальний розв'язок можна тільки за допомогою методів математичного програмування. Класифікація математичних моделей може бути проведена на основі математичних методів, до яких слід віднести [72-76, 96, 165] методи варіаційного аналізу, методи розв'язання задач комбінаторної і дискретної оптимізації, методи теорії ігор і теорії статистичних розв'язків. Основним етапом прийняття рішення є його обґрунтування на основі перерахованих математичних методів. При ухваленні рішення необхідність у розв'язанні оптимізаційних задач виникає практично на всіх етапах прийняття рішення: при постановці задачі, при побудові самої моделі та її оптимізації, при експертних оцінках отриманих розв'язків. Тому оперативність ухвалення рішення істотно залежить від автоматизації процесу розв'язання

оптимізаційних задач і вимагає організації обчислень практично в прискореному масштабі часу.

При побудові інтелектуальних мереж широке застосування знаходять математична логіка і математичне числення, що включають у себе числення висловлювань, числення задач, числення предикатів. Як показано в роботі [162], основними формальними моделями дедуктивних систем є завдання «виконуваність», «3-виконуваність», «узагальнена виконуваність», «виконуваність булевих виразів, булевих формул з кванторами», «відсутність тавтології» та ін. Особливо можна виділити завдання «виконуваність» і «3-виконуваність», які є основними формальними моделями розв’язання задач теорії побудови автоматів і мов: «допустимість кінцевим автоматом», «допустимість автоматом квазіреальності часу», «мінімальний кінцевий автомат», «покриття Рейнгольда для безконтекстних граматик», «приналежність мови квазіреальності часу» та ін. Крім проаналізованих моделей, досить широке застосування при обґрунтуванні розв’язків знаходять варіаційні моделі [51-53, 117, 298-300], зокрема задача оптимального управління [133] і задачі теорії розкладів, в основі розв’язання яких лежить задача розв’язання діофантових рівнянь з булевими змінними [134, 201, 313]. Слід зазначити, що до цього класу задач належать задачі управління порядком виконання операцій. У роботах [120, 121] показано, що широкий клас задач теорії розкладів зводиться до відомих задач теорії графів, а саме «задача про розбиття», «задача про вершинне покриття», «задача про кліки». Багато задач синтезу мереж пов'язані з розв’язанням задач оптимального розфарбовування графів і визначення незалежних максимальних множин у графах [161, 163-165]. При розв’язанні задач діагностики і контролю в мережах часто виникають задачі визначення ізоморфізму графів і підграфів, також задачі про мінімальне покриття. Крім того, питання синтезу баз даних і управління базами даних, як показано в роботах [245-247, 258, 265, 270-276], пов'язані з розв'язанням задач нелінійного булевого програмування. Для забезпечення необхідної оперативності при прийнятті рішень потрібно розроблення ПВС, що забезпечують розв’язання перерахованих вище задач великої розмірності в масштабі реального часу.

Вправи

1. Поясніть, у яких режимах здійснюється управління конфігурацією мережі і що в себе включає аналіз продуктивності мережі і управління безпекою мережі.

2. Поясніть, які аспекти функціонування мережі можуть бути стандартизовані в рамках схеми «менеджер-агент».

3. Перерахуйте основні формальні моделі задач управління телекомунікаційними системами та мережами і основні показники якості їх функціонування.

4. Чим визначається цикл оновлення інформації в телекомунікаційних системах і мережах?

5. Поясніть поняття оптимального маршруту в мережі і перерахуйте основні критерії, за якими можна визначати оптимальні маршрути.

6. Які існують стандарти систем управління на основі протоколу SNMP?

7. Поясніть, як задача визначення шляху з максимальною надійністю може бути зведена до задачі визначення найкоротших шляхів.

РОЗДІЛ 2. Основні поняття теорії графів і алгоритмів

2.1. Загальні відомості про алгоритми

З алгоритмами, тобто ефективними процедурами, які однозначно призводять до результату, математика мала справу завжди. Шкільні методи множення «стовпчиком» і ділення «кутом», метод виключення невідомих при розв’язанні системи лінійних рівнянь, правило диференціювання складної функції, спосіб побудови трикутника за трьома заданим сторонам – все це алгоритми. Однак поки математика мала справу в основному з числами і обчисленнями і поняття алгоритму ототожнювалося з поняттям методу обчислення, потреби у вивченні самого цього поняття не виникало. Традиції організації обчислень складалися століттями і стали складовою частиною загальної наукової культури так само, як і елементарні навички логічного мислення. Все різноманіття обчислень можна комбінувати з 10–15 чітко визначених операцій арифметики, тригонометрії та аналізу. Тому поняття методу обчислення вважалося спочатку зрозумілим і не потребувало спеціальних досліджень.

До середини XIX століття єдиною областю математики, яка працювала з нечисловими об'єктами, була геометрія, і саме вона, не маючи можливості спиратися на обчислювальну інтуїцію людини, різко відрізнялася від решти математики підвищеними вимогами до строгості своїх міркувань. До цих пір будь–який сучасник, для якого математика – це світ обчислень (і в цьому він мало чим відрізняється від типового інженера), болісно звикає до понять доказу і ніяк не може зрозуміти, навіщо доводити рівність відрізків, коли їх простіше виміряти, і навіщо будувати перпендикуляр за допомогою циркуля і лінійки, коли є транспортир.

Таке саме болісне звикання до нових, більш жорстких вимог почалося в математиці у другій половині XIX століття. Воно стимулювалося в основному математикою нечислових об'єктів – відкриттям неевклідової геометрії, появою абстрактних алгебраїчних теорій типу і т. д. Однією з вирішальних обставин, що призвели до перегляду основ математики, тобто принципів, що лежать в основі математичних міркувань, стало створення Кантором теорії множин. Досить швидко стало зрозуміло, що поняття теорії множин у силу своєї спільності

лежить в основі всієї побудови математики. Проте майже так само швидко було показано, що не вирішено суперечності парадоксів теорії множин. Все це вимагало точного вивчення принципів математичних міркувань (досі здавалися інтуїтивно зрозумілими) математичними ж засобами.

Досвід парадоксів теорії множин навчив математику вкрай обережно поводитися з нескінченністю і по можливості навіть про нескінченність міркувати за допомогою фінітних методів. Сутність фінітного підходу полягає в тому, що він пропускає тільки кінцеві комплекси дій над кінцевим числом об'єктів. З'ясування того, які об'єкти і дії над ними слід вважати точно визначеними, якими властивостями і можливостями володіють комбінації елементарних дій, що можна і чого не можна зробити за їх допомогою, – все це стало предметом теорії алгоритмів і формальних систем, яка спочатку виникла в рамках метаматематики і стала найважливішою її частиною. Головним всередині математичним додатком теорії алгоритмів з'явилися докази неможливості алгоритмічного (тобто точного та однозначного) вирішення деяких математичних проблем. Такі докази (і точні формулювання тверджень) нездійсненні без точного поняття алгоритму.

Поки техніка використовувала чисто обчислювальні методи, ці високі проблеми чистої математики її мало цікавили. У техніку термін «алгоритм» прийшов разом з кібернетикою. Якщо поняття методу обчислення не потребувало пояснень, то поняття процесу управління довелося виробляти практично заново. Знадобилося усвідомлення, які вимоги повинні задовольняти послідовність дій (чи її опис), щоб вважатися ефективно заданою, тобто мати право називатися алгоритмом. У цьому усвідомленні величезну допомогу надала практика використання обчислювальних машин, що зробила поняття алгоритму відчутною реальністю. З точки зору сучасної практики, алгоритм – це програма, а критерієм алгоритмічності процесу є можливість його запрограмувати. Саме завдяки цій реальності алгоритму, а також тому, що підхід інженера до математичних методів завжди був конструктивним, поняття алгоритму в техніці за короткий термін стало надзвичайно популярним (навіть більше, ніж у самій математиці).

Однак у будь-якої популярності є свої витрати. У повсякденній практиці слово «алгоритм» вживається дуже широко, втрачаючи найчастіше свій точний сенс. Приблизні описи поняття «алгоритм» часто приймаються за точні визначення. У результаті за алгоритм часто видається будь–яка інструкція, розбита на кроки. З'являються такі дикі словосполучення, як «алгоритм винаходу» (адже наявність «алгоритму винаходу» означало б кінець винахідництва як творчої діяльності).

Чітке уявлення про те, що таке алгоритм, важливо, звичайно, не лише для правильного слововживання. Воно потрібно і при розробленні конкретних алгоритмів, особливо коли розуміється їхнє подальше програмування. Однак воно ще важливіше при наведенні порядку в

бурхливо зростаючому алгоритмічному господарстві. Щоб орієнтуватися в морі алгоритмів, що охопило технічну кібернетику, необхідно вміти порівнювати різні алгоритми розв’язання одних і тих самих задач, причому не тільки за якістю розв’язання, але і за характеристиками самих алгоритмів (кількість дій, витрата пам'яті і т. д.). Таке порівняння неможливо без введення точної мови для обговорення всіх цих питань, інакше кажучи, самі алгоритми повинні стати такими самими предметами точного дослідження, як і ті об'єкти, для роботи з якими вони призначені.

Дослідження основних понять теорії алгоритмів почнеться в п. 2.2. Перш ніж приступити до нього, обговоримо на неформальному рівні деякі основні принципи, за якими будуються алгоритми, з'ясуємо, що ж саме в понятті алгоритму потребує уточнення.

Основні вимоги до алгоритмів

1. Перше, що слід відзначити в будь–якому алгоритмі, – це те, що він застосовується до вихідних даних і видає результати. У звичних технічних термінах це означає, що алгоритм має входи і виходи. Крім того, під час роботи алгоритму з'являються проміжні результати, які використовуються в подальшому. Таким чином, кожен алгоритм має справу з вхідними, проміжними і вихідними даними. Оскільки ми збираємося уточнювати поняття алгоритму, потрібно уточнити і поняття даних, тобто вказати, які вимоги повинні задовольняти об'єкти, щоб алгоритми могли з ними працювати,

Зрозуміло, що ці об'єкти повинні бути чітко визначені і відмінні як один від одного, так і від «необ'єктів». У багатьох важливих випадках добре відомо, що це означає: до таких алгоритмічних об'єктів належать числа, вектори, матриці суміжності графів, формули. Зображення (наприклад, рисунок графа) є менш природними, ніж алгоритмічні об'єкти. Якщо говорити про графи, то справа навіть не в тому, що в рисунку більше несуттєвих деталей і дві людини один і той самий граф зобразять по-різному (зрештою, різні матриці суміжності теж можуть задавати один і той самий граф з точністю до ізоморфізму), а в тому, що матриця суміжності легко розбивається на елементи, причому з елементів лише двох видів (нулів і одиниць) складаються матриці будь–яких графів, тоді як розбити на елементи рисунок набагато важче. Нарешті, з такими об'єктами, як «хороша книга» або «осмислене твердження», з якими легко управляється будь–яка людина (але кожна по–своєму!), алгоритм працювати відмовиться, поки вони не будуть описані як дані за допомогою інших, більш придатних об'єктів.

Замість того, щоб намагатися дати загальне словесне визначення чіткої визначеності об'єкта, у теорії алгоритмів фіксують конкретні кінцеві набори початкових об'єктів (елементарних) і кінцевий набір засобів побудови інших об'єктів з елементарних. Набір елементарних об'єктів утворює кінцевий алфавіт вихідних символів (цифр, букв і т. д.), з яких будуються інші об'єкти; типовим засобом побудови є індуктивні

визначення, що вказують, як будувати нові об'єкти з вже побудованих. Найпростіше індуктивне визначення – це визначення деякої безлічі слів, класичним прикладом якого служить визначення ідентифікатора в мові програмування: ідентифікатор – це або буква, або ідентифікатор, до якого приписана праворуч буква або цифра. Слова кінцевої довжини в кінцевих абетках (зокрема, числа) – найбільш звичайний тип алгоритмічних даних, а кількість символів у слові (довжина слова) – природна одиниця вимірювання об'єму оброблюваної інформації. Більш складний випадок алгоритмічних об'єктів – формули. Вони також визначаються індуктивно і також є словами в кінцевому алфавіті, проте не кожне слово в цьому алфавіті є формулою. У цьому випадку зазвичай основним алгоритмам передують допоміжні, які перевіряють, чи задовольняють вихідні дані потрібні вимоги. Така перевірка називається синтаксичним аналізом.

2. Дані для свого розміщення вимагають пам'яті. Пам'ять зазвичай вважається однорідною і дискретною, тобто складається з однакових клітинок, причому кожна клітинка може містити один символ алфавіту даних. Таким чином, одиниці вимірювання об'єму даних і пам'яті узгоджені. При цьому пам'ять може бути нескінченною. Питання про те, чи потрібна одна пам'ять або декілька і, зокрема, чи потрібна окрема пам'ять для кожного з трьох видів даних, вирішується по–різному.

3. Алгоритм складається з окремих елементарних кроків, або дій. Типовий приклад безлічі елементарних дій – система команд ЕОМ. Зазвичай елементарний крок має справу з фіксованою кількістю символів (це зручно, наприклад, для вимірювання часу роботи алгоритму кількістю виконаних кроків), однак ця вимога не завжди виконується. Наприклад, в ЕОМ третього покоління є команди типу «пам'ять – пам'ять», що працюють з полями пам'яті змінної довжини.

4. Послідовність кроків алгоритму детермінована, тобто після кожного кроку або вказується, який крок робити далі, або дається команда зупинки, після чого робота алгоритму вважається закінченою.

5. Природно від алгоритму зажадати результативності, тобто зупинки після кінцевої кількості кроків (що залежить від даних) із зазначенням того, що вважати результатом. Зокрема, кожен, хто пред'являє алгоритм розв'язання деякої задачі, наприклад обчислення функції $f(x)$, зобов'язаний показати, що алгоритм зупиняється після кінцевої кількості кроків (як кажуть, збігаються) для будь–якого x з області задавання f. Однак перевірити результативність (збіжність) набагато важче, ніж вимоги, викладені в п. 1 – 4. На відміну від них, збіжність зазвичай не вдається встановити простим переглядом опису алгоритму; загального ж методу перевірки збіжності, придатного для будь–якого алгоритму A і будь–яких даних взагалі не існує.

6. Слід розрізняти: а) опис алгоритму (інструкцію або програму); б) механізм реалізації алгоритму (наприклад, ЕОМ), що включає кроки пуску, зупинки, реалізації елементарних кроків, видачі результатів і

забезпечення детермінованості, тобто управління ходом підрахунку; в) процес реалізації алгоритму, тобто послідовність кроків, яка буде породжена при застосуванні алгоритму до конкретних даних. Будемо припускати, що опис алгоритму і механізм його реалізації кінцеві (пам'ять, як уже зазначалося, може бути нескінченною, але вона не включається в механізм). Вимоги до закінчення процесу реалізації збігаються з вимогами результативності (див. п. 5).

Приклад 2.1. Розглянемо таку задачу. Дано послідовність P з n додатних чисел (n – кінцеве, але довільне число); потрібно впорядкувати їх, тобто побудувати послідовність R, у якій ці самі числа розташовані в порядку зростання. Майже відразу ж спадає на думку такий простий спосіб її розв’язання: переглядаємо P і знаходимо в ній найменше число; викреслюємо його з P і виписуємо його в якості першого числа R, знову повертаємося до P і знаходимо в ній найменше число; приписуємо його праворуч до одержаної частини R і так далі, до тих пір, поки в P не будуть викреслені всі числа.

Виникає природне запитання: що означає «і так далі»? Для більшої зрозумілості перепишемо опис способу розв’язання у більш чіткій формі, розбивши його на кроки і вказавши переходи між кроками.

Крок 1. Шукаємо в P найменше число.

Крок 2. Знайдене число приписуємо праворуч до R (у початковий момент R порожня) і викреслюємо його з P.

Крок 3. Якщо в P немає чисел, то переходимо до кроку 4. В іншому випадку переходимо до кроку 1.

Крок 4. Кінець. Результатом вважати послідовність R, побудовану до даного моменту.

Більшість читачів вважатиме такий опис досить зрозумілим (і навіть зайво формальним), щоб, користуючись ним, однозначно отримати потрібний результат. Проте це враження зрозумілості спирається на деякі неявні припущення, до правильності яких ми звикли, але які неважко порушити. Наприклад, що означає «дана послідовність чисел»? Чи є така послідовність 3 (1 / 7), 2 (1 / 5), (1, 2)? Очевидно, так, однак у нашому описі нічого не сказано, як знайти найменше число серед таких чисел. У ньому взагалі не йдеться про те, як шукати найменші числа, мабуть, передбачається, що йдеться про числа, представлені у вигляді десяткових дробів, і що відомо, як їх порівнювати.

Отже, необхідно уточнити форми представлення даних. При цьому не можна просто заявити, що припустимим є будь-яке подання чисел. Адже для кожного подання існує свій алфавіт (який, крім цифр, може включати коми, дужки, знаки операцій і функцій і т. д.) і свій спосіб порівняння чисел (наприклад, спосіб переведення в десятковий дріб), тоді як кінець алфавіту вимагає фіксувати його заздалегідь, а кінець опису алгоритму дозволяє включити в нього лише заздалегідь фіксовану кількість способів порівняння. Фіксація подання чисел у вигляді десяткових дробів також не

вирішує всіх проблем. Порівняння 10–20–розрядних чисел вже не може вважатися елементарною дією: відразу не можна сказати, яке з чисел більше: 90811557001,15 або 32899901467,0048. У машинних алгоритмах саме представлення числа ще вимагає подальшого уточнення: потрібно, по–перше, обмежити кількість розрядів (цифр) у числі, оскільки від цього залежить, скільки елементів пам'яті буде займати число, а по-друге, домовитися про спосіб розміщення десяткової коми в числі, тобто вибрати подання у вигляді фіксованої або рухомої коми, оскільки способи обробки цих двох подань різні. Нарешті, кожен, хто мав справу з програмуванням, відзначить, що на кроці 1 потрібно дізнатися дві речі: найменше число (щоб записати його в *R*) і його місце в *P*, тобто його адресу в тій частині пам'яті, де зберігається *P* (щоб викреслити його з *P*), а отже, потрібно мати вказівки цієї адреси.

Таким чином, навіть у цьому простому прикладі нескладний аналіз показує, що опису, який виглядає цілком зрозумілим, ще далеко до алгоритму. Ми зіткнулися тут з необхідністю уточнити майже всі основні характеристики алгоритму, які відзначалися раніше: алфавіт даних і форму їх подання, пам'ять і розміщення в ній елементів *P* і *R*, елементарні кроки (оскільки крок 1 явно неелементарний). Крім того, стає зрозумілим, що вибір механізму реалізації (скажімо, людина або ЕОМ) буде впливати і на сам характер уточнення: у людини вимоги до пам'яті, представлення даних і до елементарності кроків набагато слабкіші, окремі незначні деталі вона може уточнити сама.

Про підходи до уточнення поняття «алгоритм»

Раніше були сформульовані основні вимоги до алгоритмів. Однак поняття, використані в цих формулюваннях (такі, як ясність, чіткість, елементарність), самі потребують уточнення. Очевидно, що їхні словесні визначення будуть містити нові поняття, які знову потребують уточнення, і т. д. Тому в теорії алгоритмів приймається інший підхід: вибирається кінцевий набір вихідних об'єктів, які оголошуються елементарними, і кінцевий набір способів побудови з них нових об'єктів. Цей підхід був уже використаний при обговоренні питання про дані: уточненням поняття «дані» надалі будемо вважати множини слів у кінцевих абетках. Для уточнення детермінізму використовуватимуться або блок–схеми та еквівалентні їм словесні описи (типу того, який наведений у прикладі, або опис механізму реалізації алгоритму). Крім того, потрібно зафіксувати набір елементарних кроків і домовитися про організацію пам'яті. Після того як це буде зроблено, вийде конкретна алгоритмічна модель.

Алгоритмічні моделі, що розглядаються в цьому розділі, претендують на право вважатися формалізацією поняття «алгоритм». Це означає, що вони повинні бути універсальними, тобто допускати опис будь-яких алгоритмів. Тому може виникнути природне заперечення проти пропонованого підходу: чи не призведе вибір конкретних засобів до втрати спільності формалізації? Якщо мати на увазі основні цілі, що стояли при

створенні теорії алгоритмів, – універсальність і пов'язана з нею можливість говорити в рамках будь–якої моделі про властивості алгоритмів взагалі, то це заперечення знімається в такий спосіб. По-перше, доводиться зводити одні моделі до інших, тобто показується, що будь-який алгоритм, описаний засобом однієї моделі, може бути описаний засобами іншої. По–друге, завдяки взаємному зведенню моделей у теорії алгоритмів вдалося виробити інваріантну відносно моделей систему понять, що дозволяє говорити про властивості алгоритмів незалежно від того, яку формалізацію алгоритму обрано. Ця система понять заснована на понятті обчислюваної функції, тобто функції, для обчислення якої існує алгоритм.

Тим не менше, хоча спільність формалізації в конкретній моделі не втрачається, різний вибір вихідних витрат призводить до моделей різного виду. Можна виділити три основні типи універсальних алгоритмічних моделей, що розрізняються вихідними евристичними міркуваннями щодо того, що таке алгоритм. Перший тип пов'язує поняття алгоритму з найбільш традиційними поняттями математики – обчисленнями і числовими функціями. Найбільш розвинена і вивчена модель цього типу – рекурсивні функції - є історично першою формалізацією поняття алгоритму. Другий тип, заснований на уявленні про алгоритм як про детермінований пристрій, здатен виконувати в кожний окремий момент лише досить примітивні операції. Таке уявлення не залишає сумнівів в однозначності алгоритму і елементарності його кроків. Крім того, евристика цих моделей близька до ЕОМ і, отже, до інженерної інтуїції. Основною теоретичною моделлю цього типу (створену в 1930–х роках раніше за ЕОМ!) є машина Тюрінга. Нарешті, третій тип алгоритмічних моделей – це перетворення слів у довільних символах, у яких елементарними операціями є підстановки, тобто заміни частки слова (підслова) іншим словом. Переваги цього типу – у його максимальній абстрактності і можливості застосувати поняття алгоритму до об'єктів довільної (не обов'язково числової) природи. Втім, як буде зрозуміло в подальшому, моделі другого і третього типу досить близькі (їх взаємність доводиться просто) і відрізняються в основному евристичними акцентами. Приклади моделей цього типу – канонічні системи Посту і нормальні алгоритми Маркова.

Формалізація поняття алгоритму на основі машини Тюрінга

Для формалізації поняття алгоритму і його складності необхідно вибрати деяку конкретну модель обчислень. Зручною моделлю є так звані машини Тюрінга. Попередньо введемо кілька допоміжних понять.

Алфавітом називається довільна кінцева безліч символів, званих літерами. Слово в даному алфавіті – це кінцева непуста послідовність букв алфавіту. Під довжиною слова розуміється кількість букв, що входять до нього (кожна літера враховується стільки разів, скільки вона входить у слово).

Детермінована машина Тюрінга (ДМТ) складається зі стрічки, управляючого пристрою і зчитувальної головки.

Стрічка розділена на ***клітинки*** і потенційно нескінченна в обидва боки. Клітинки пронумеровані числами ..., –2, –1, 0, 1, 2, ... Будь–яка клітинка стрічки може знаходитися в одному з кінцевої кількості станів, кожному з яких поставлена у взаємно–однозначну відповідність буква алфавіту B (званого зовнішнім алфавітом). Виділяється буква $b_0 \in B$ – ***порожній символ***.

Управляючий пристрій у кожний момент часу знаходиться в одному з кінцевої кількості станів, що позначається літерами внутрішнього алфавіту D і званих внутрішніми станами машини. Зазначимо, що $B \cap D = \varnothing$. Виділяються два особливих стани: початковий, що позначається літерою $d_н$, і завершальний, що позначається літерою $d_з$.

Зчитувальна головка машини може пересуватися вздовж стрічки і в кожен момент часу оглядати рівно одну клітинку стрічки. Зчитувальна головка може сприймати стан розглянутих клітинок і замінювати один стан іншим. Зазвичай виділяється вхідний алфавіт B^*, що є підмножиною множини B. Зокрема $b_0 \notin B$. Один крок роботи ДМТ полягає у виконанні залежно від стану управляючого пристрою та стану розглянутих стрічки всіх або деяких з дій:

1) змінити внутрішній стан машини;

2) змінити стан розглянутих клітинок стрічки;

3) зрушити зчитувальну головку на одну клітинку вліво (L) або на одну клітинку вправо (P), або залишити її на місці (S).

Далі не будемо робити відмінності між станом клітинки стрічки (станом упавляючого пристрою) і відповідною цьому стану буквою алфавіту B (відповідно алфавіту D).

Як математичний об'єкт ДМТ визначається набором (B, B^*, D, b_0, $d_н$, $d_з$, δ). Тут δ – відображення деякої непорожньої підмножини множини B'D (що не містить пар виду ($d_з$, d_i), де $d_i \in D$) у безліч $B \times D \times \{L, P, S\}$. Відображення δ називається функцією переходів.

Станом ДМТ називається сукупність, що складається:

а) із послідовності $b_{i_1}, b_{i_2} \ldots b_{i_p}$ станів всіх клітинок стрічки, $b_{i_r} \in B^* \left(r = \overline{1, p}\right)$ (всі клітинки, розташовані ліворуч від комірки, що знаходиться в стані b_{i_1}, і праворуч від комірки, що знаходиться в стані b_{i_p}, порожні і послідовно пропущені $b_{i_1} \neq b_0, b_{i_p} \neq b_0$);

б) внутрішнього стану $d \in B$, у якому в даний момент часу знаходиться управляючий пристрій машини;

в) стану b_{i_k} розглянутих комірок стрічки. Стан ДМТ в кожен момент часу однозначно визначається миттєвим описом–словом $b_{i_1} b_{i_2} \ldots\ldots d b_{i_k} \ldots b_{i_p}$ в алфавіті $D \cup B$. Тут символ $d \in B$ розташований перед символом, який

показує стан розглянутих у даний момент комірок стрічки. Стан ДМТ, що визначається миттєвим описом виду $b_{i_1} b_{i_2} \ldots\ldots db_{i_k} \ldots b_{i_p}$, називається заключним (управляючий пристрій знаходиться в завершальному стані $d_з$). Потрапивши до завершального стану, машина зупиняється.

Поняття алгоритму, подібно поняттю множини, належить до числа понять настільки фундаментальних, що не може бути виражене через інші, а має розглядатися як таке, що не визначається. Можна лише дати пояснення такого типу: «Алгоритм – це точне розпорядження, яке задає обчислювальний процес, що починається з довільно обраного початкового даного і спрямований на отримання повністю визначуваного цим вихідним даним результату». З точки зору сучасної практики алгоритм – це програма, а критерієм алгоритмічності процесу є можливість його запрограмувати. Ми при аналізі алгоритмів час роботи алгоритму будемо виражати в термінах кількості елементарних кроків (арифметичних операцій, порівнянь, команд розгалуження і т. д.), при цьому будемо припускати, що всі ці операції вимагають одну одиницю часу. Кількість кроків, які виконує алгоритм для різноманітних вихідних даних, не однакова. Тому необхідно розглядати всі входи даного розміру n разом. Визначимо складність алгоритму для цього розміру входу як кількість кроків алгоритму в гіршому випадку по всіх цих входах. Тоді складність алгоритму можна представляти у вигляді функції розміру входу, такої як $10n^2$, 2^n і т. д. Відмінності між алгоритмами складності $10n^2$ і $9n^2$ дають багаторазове збільшення швидкості обчислювальних машин. З іншого боку, якщо складність визначається функцією n^3+n^2, то повільно зростаючі складові будуть поглинатися більш швидко зростаючими, так $n^3 (1 + 4n+1) \approx n^3$ доданок n^2 поглинається доданком n^3.

Швидкість зростання функцій. Нехай $f(n)$, $g(n)$ – функції визначені на множині цілих додатних чисел і приймають додатні дійсні значення. Введемо такі позначення:

а) $f(n) = O(n)$, якщо існує така константа $C > 0$, що при $n > n_0$ виконується нерівність $f(n) \le C\, g(n)$;

б) $f(n) = \Omega(n)$, якщо існує така константа $C > 0$, що при $n > n_0$ виконується нерівність $f(n) \ge C\, g(n)$;

в) $f(n) = \Theta(n)$, якщо існують такі константи C_1, $C_2 > 0$, що при $n > n_0$ виконується рівність $f(n) = C\, g(n)$.

На рис. 2.1 наведені ілюстрації швидкості зростання функцій для випадків а); б); в).

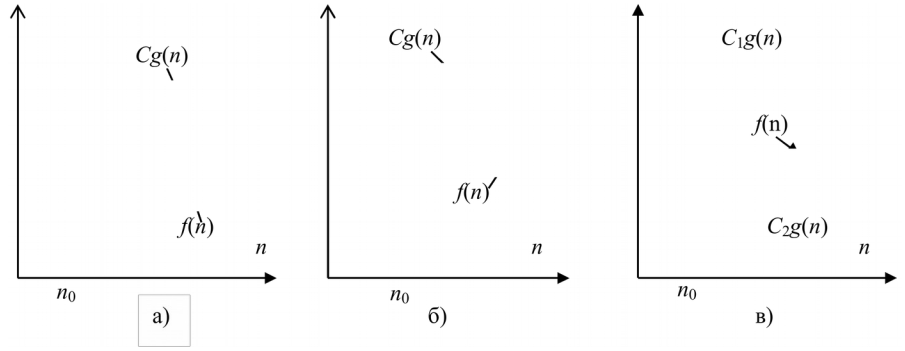
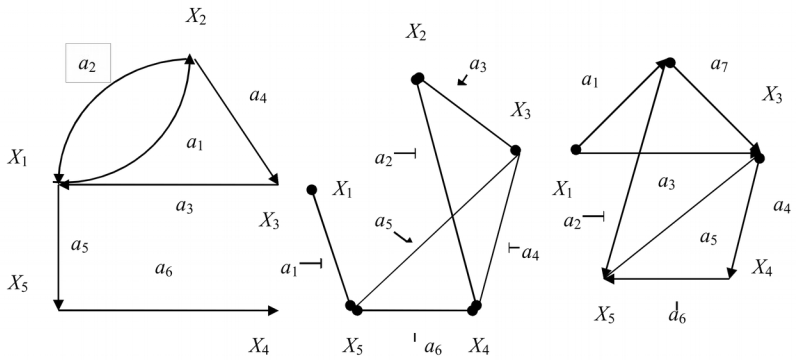


Рис. 2.1. Швидкість зростання функцій

Таким чином, безліч всіх функцій $g\ (n)$ таких, що $f\ (n) = \Theta\ (n)$, називається швидкістю зростання $f\ (n)$, а записи а) і б) визначають верхню та нижню оцінки зростання швидкості. Завдяки введенню поняття швидкість зростання складності алгоритму можна оцінити зверху, використовуючи вирази типу «вимагає часу $O\ (n^3)$)».

2.2. Основні поняття теорії графів

Граф G задається множиною точок або вершин x_1, x_2, ... , x_n (яке позначається через X) і безліччю ліній або ребер a_1, a_2, ... , a_m (яке позначається символом A), що сполучають між собою всі або частину цих точок. Таким чином, граф G повністю задається (і позначається) парою (X, A). Якщо ребра з безлічі A орієнтовані, що зазвичай показується стрілкою, то вони називаються дугами, і граф з такими ребрами називається орієнтованим графом (рис. 2.2, а). Якщо ребра не мають орієнтації, то граф називається неорієнтованим (рис. 2.2, б).



а б в

Рис. 2.2. Графи: а) орієнтований; б) неорієнтований; в) змішаний

Коли дуга позначається впорядкованою парою, що складається з початкової та кінцевої вершин (тобто двома кінцевими вершинами дуги), її напрямок передбачається заданим від першої вершини до другої. Якщо x_1 і x_2 – кінцеві вершини дуги a, то говорять, що вершини x_1 і x_2 інцидентні дузі

а (або дуга а інцидентна вершинам x_1 і x_2). Так, наприклад, на рис. 2.2, а позначення (x_1, x_2) належить до дуги a_1, а (x_2, x_1) – до дуги a_2.

Вживається часто опис орієнтованого графа G в задаванні безлічі вершин X та відповідності Γ, яке показує, як між собою пов'язані вершини. Відповідність Γ називається відображенням множини X в X, а граф у цьому випадку позначається парою $G = (X, \Gamma)$.

Для графа на рис. 2.2, а маємо $\Gamma(x_1) = (x_2, x_5)$, тобто вершини x_2 і x_5 є кінцевими вершинами дуг, у яких початковою вершиною є x_1, $\Gamma(x_2) = (x_1, x_3)$; $\Gamma(x_3) = (x_1)$; $\Gamma(x_4) = \varnothing$ – порожня множина; $\Gamma(x_5) = (x_4)$.

У разі неорієнтованого графа чи графа, що містить і дуги, і неорієнтовані ребра (наприклад, графи, зображені на рис. 2.2, б та 2.2, в), передбачається, що відповідність Γ задає такий еквівалентний орієнтований граф, який виходить з вихідного графа заміною кожного неорієнтованого ребра двома протилежно спрямованими дугами, що з'єднують ті самі вершини. Так, наприклад, для графа, наведеного на рис. 2.2, б, маємо $\Gamma(x_5) = \{x_1, x_3, x_4\}$, $\Gamma(x_1) = \{x_5\}$ і т. ін.

Оскільки $\Gamma(x_i)$ являє собою безліч таких вершин $x_j \in X$, для яких у графі G існує дуга (x_i, x_j), то через $\Gamma^{-1}(x_i)$ природно позначити безліч вершин x_k, для яких у G існує дуга (x_k, x_i). Відношення $\Gamma^{-1}(x_i)$ прийнято називати зворотною відповідністю. Для графа, зображеного на рис. 2.2, а, маємо $\Gamma^{-1}(x_1) = \{x_2, x_3\}$; $\Gamma^{-1}(x_2) = \{x_1\}$ і т. ін.

Цілком очевидно, що для неорієнтованого графа $\Gamma^{-1}(x_i) = \Gamma(x_i)$ для всіх $x_i \in X$.

Коли відображення Γ діє не на одну вершину, а безліч вершин $X_q = (x_1, x_2, ..., x_q)$, то під $\Gamma(X_q)$ розуміють об'єднання $\Gamma(x_1) \cup \Gamma(x_2) \cup ... \cup \Gamma(x_q)$,

тобто $\Gamma(X_q)$ є безліччю таких вершин $x_j \in X$, що для кожної з них існує дуга (x_i, x_j) у G, де $x_i \in X_q$. Для графа, наведеного на рис. 2.2, а, $\Gamma(\{x_2, x_5\}) = \{x_1, x_3, x_4\}$ і $\Gamma(\{x_1, x_3\}) = \{x_2, x_5, x_1\}$.

Відображення $\Gamma(\Gamma(x_i))$ записується як $\Gamma^2(x_i)$. Аналогічно «потрійне» відображення $\Gamma(\Gamma(\Gamma(x_i)))$ записується як $\Gamma^3(x_i)$ і т. д. Для графа, показаного на рис. 2.2, а, маємо:

$$\Gamma^2(x_1) = \Gamma(\Gamma(x_1)) = \Gamma(\{x_2, x_5\}) = \{x_1, x_3, x_4\},$$
$$\Gamma^3(x_1) = \Gamma(\Gamma^2(x_1)) = \Gamma(\{x_1, x_3, x_4\}) = \{x_1, x_2, x_5\} \text{ і т. ін.}$$

Аналогічно розуміються позначення $\Gamma^{-2}(x_i)$, $\Gamma^{-3}(x_i)$ і т. ін.

Шляхи та маршрути

Шляхом (або орієнтованим маршрутом) орієнтованого графа називається послідовність дуг, у якій кінцева вершина будь-якої дуги, відмінної від останньої, є початковою вершиною наступної.

Так, на рис. 2.2, в послідовності дуг $a_1, a_7, a_4, a_6, a_1, a_2, a_5, a_4, a_3, a_4, a_6$ є шляхами.

Дуги $a = (x_i, x_j)$, $x_i \neq x_j$, що мають спільні кінцеві вершини, називаються суміжними. Дві вершини x_i і x_j називаються суміжними, якщо яка-небудь з двох дуг (x_i, x_j) і (x_j, x_i) або обидві одночасно присутні в графі. Так, наприклад, на рис. 2.2, в дуги a_1, a_2 і a_7, як і вершини x_2 і x_3, є суміжними, у той час як дуги a_1 і a_4 або вершини x_1 і x_4 не є суміжними.

Орієнтованим ланцюгом називається такий шлях, у якому кожна дуга використовується не більше одного разу. Так, наприклад, наведені вище шляхи є орієнтованими ланцюгами.

Простим орієнтованим ланцюгом називається такий шлях, у якому кожна вершина використовується не більше одного разу. Маршрут є неорієнтованим «двійником» шляху, і це поняття розглядається в тих випадках, коли можна знехтувати спрямованістю дуг у графі.

Ваги й довжина шляхів

Іноді дугам графа G приписуються числа дуги (x_i, x_j), тобто ставиться у відповідність деяке число c_{ij}, називане вагомим, або довжиною, або вартістю (ціною) дуги. Тоді граф G називається графом зі зваженими дугами. Іноді ваги (числа v_i) приписуються вершинам x_i графа, і тоді утворюється граф зі зваженими вершинами. Якщо в графі ваги приписані й дугам, і вершинам, то він називається просто зваженим. При розгляді шляху μ, представленого послідовністю дуг $(a_1, a_2, ..., a_q)$, за його вагу (або довжину, або вартість) приймається число $l(\mu)$, що дорівнює сумі ваг всіх дуг, що входять у μ, тобто $l(\mu) = \sum_{(x_i, x_j) \in \mu} c_{ij}$. Таким чином, коли слова «довжина», «вартість», «ціна» і «вага» застосовуються до дуг, то вони еквівалентні за змістом, і в кожному конкретному випадку вибирається таке слово, що ближче підходить за змістом і збігається з прийнятим у літературі. Довжиною (або рангом) шляху називається кількість дуг, що входять у нього.

Петлі, орієнтовані цикли й цикли

Петлею називається дуга, початкова й кінцева вершини якої збігаються. На рис. 2.3, наприклад, дуги a_2 і a_5 є петлями.

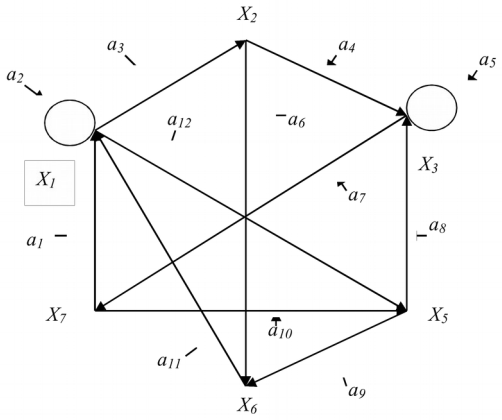


Рис. 2.3. Петлі й цикли

Шлях a_1, a_2, ..., a_q називається замкнутим, якщо в ньому початкова вершина дуги a_1 збігається з кінцевою вершиною дуги a_q.

Так, наприклад, у графі, наведеному на рис. 2.3, послідовності дуг

$$a_3, a_6, a_{11}, \tag{2.1}$$

$$a_{11}\ a_3\ a_4\ a_7\ a_1\ a_{12}\ a_9, \tag{2.2}$$

$$a_3\ a_4\ a_7\ a_{10}\ a_9\ a_{11} \tag{2.3}$$

є замкнутими шляхами.

Шляхи (2.1) і (2.3) є замкнутими простими циклами (контурами), оскільки в них та сама вершина використовується тільки один раз (за винятком початкової й кінцевої вершин, які збігаються), а шлях (2.2) не є контуром, тому що вершина x_1 використовується в ньому двічі. Контур, що проходить через всі вершини графа, має особливе значення й називається гамільтоновим контуром. Звичайно, не всі графи мають гамільтонові контури. Так, наприклад, контур (2.3) є гамільтоновим контуром графа, наведеного на рис. 2.3, а граф на рис. 2.2, а не має гамільтонових контурів.

Замкнутий маршрут є неорієнтованим двійником замкнутого шляху. Таким чином, замкнутий маршрут є маршрутом x_1, x_2, ..., x_1, у якому збігаються початкова й кінцева вершини. На рис. 2.3 маршрут a_1, a_{12}, a_{10} є замкнутим маршрутом.

Ступені вершини

Кількість дуг, які мають вершину x_i своєю початковою вершиною, називається напівступенем результату вершини x_i, і, аналогічно, кількість дуг, які мають x_i своєю кінцевою вершиною, називається напівступенем заходу вершини x_i.

Зовсім очевидно, що сума напівступенів заходу всіх вершин графа, а також сума напівступенів результату всіх вершин дорівнюють загальній кількості дуг графа G, тобто $\sum_{i=1}^{n} d_-(x_i) = \sum_{i=1}^{n} d_+(x_i) = m$, де n – кількість вершин; m – кількість дуг графа G.

Для неорієнтованого графа G – (X, Γ) ступінь вершини X_i визначається аналогічно – за допомогою співвідношення $d(x_i) \equiv |\Gamma(x_i)|$, і коли не може виникнути непорозумінь, ми будемо позначати ступінь вершини x_i через d_i. Для довільних неорієнтованих графів справедливі такі твердження.

***Твердження* 2.1.** Сума ступенів вершин будь–якого графа - парне число, що дорівнює подвоєній кількості його ребер.

Нехай x_1, x_2, ..., x_n вершини деякого графа, а $d(x_1)$, $d(x_2)$, …, $d(x_n)$ – ступені цих вершин. Підрахуємо кількість ребер, що сходяться в кожній вершині, і просумуємо ці числа. Це рівносильно знаходженню суми ступенів всіх вершин. При такому підрахунку кожне ребро буде враховано двічі. Звідси

$$\sum_{i=1}^{n} d(x_i) = 2m,$$

де m – кількість ребер.

Твердження 2.2. У будь–якому графі кількість вершин непарного ступеня є парною.

Нехай $a_1, a_2, \ldots, a_к$ – ступені вершин з парними ступенями, а $b_1, b_2, \ldots, b_p$ – ступінь вершин з непарними ступенями.

Сума $a_1 + a_2 + \ldots + a_к + b_1 + b_2 + \ldots + b_p$ рівно удвічі перевищує кількість ребер графа. Сума $a_1 + a_2 + \ldots + a_к$ – парна (сума парних чисел), тоді сума $b_1 + b_2 + \ldots + b_p$ повинна бути парною. Це можливо лише в тому випадку, якщо p – парне, тобто «парним» є кількість непарних вершин графа.

Твердження 2.3. У будь-якому графі з n вершинами, де $n \geq 2$, завжди найдуться дві або більше вершини з однаковими ступенями.

Якщо граф має n вершин, то кожна з них може мати ступінь 0, 1, 2,…, n–1... . Припустимо, що існує граф, де різні ступені вершин, тобто $d(x_i)$, приймають значення з безлічі чисел 0, 1, 2, …, n – 1, але якщо є вершина x_A зі ступенем 0, то це означає, що вона ізольована й, отже, у графі не знайдеться вершини зі ступенем (n – 1), оскільки ця вершина повинна бути з'єднана з (n – 1)–ю вершиною, у тому числі й з (x_A, x_A), яка виявилася ізольованою. Отже, у графі, що має n вершин, не можуть бути одночасно вершини ступеня 0 і (n – 1). Це означає, що з n вершин найдуться дві, що мають однакові ступені.

Завдання: довести, що якщо у футбольному турнірі беруть участь 20 команд, то в будь–який момент часу є дві команди, що зіграли однакову кількість матчів.

Наслідком твердження 2.3 є наступне твердження.

Твердження 2.4. Якщо в графі з n вершинами при $n \geq 2$ тільки одна пара має однаковий ступінь, то в цьому графі завжди знайдеться або єдина ізольована вершина, або єдина вершина, з'єднана з усіма іншими.

Завдання для підтвердження твердження 2.4:

серед n абонентів мережі, що обмінювалися повідомленнями, у деякий момент з'ясувалося, що двоє абонентів зробили однакову кількість обмінів.

Довести що серед абонентів є або один, що ще не почав обміну, або один, що уже завершив обміни з усіма абонентами.

Підграфи

Нехай задано граф $G = (X, A)$. Остовим підграфом G_p графа G називається граф (X, A_p), для якого $A_p \subset$ A. Таким чином, остовий підграф має ту саму безліч вершин, що й граф G, але безліч дуг підграфа G_p є підмножиною безлічі дуг вихідного графа.

Граф на рис. 2.4, б – остовий підграф G_p графа G, зображеного на рис. 2.4, а.

Нехай задано граф $G = (X, \Gamma)$. Породженим підграфом G_S називається граф (X_S, Γ_S), для якого $X_S \subset X$ і для кожної вершини $x_i \in X_S$, $\Gamma(x_i) = \Gamma(x_i) \cap X_S$.

Таким чином, породжений підграф складається з підмножини вершин X_S, безлічі вершин вихідного графа й всіх таких дуг графа G, у яких кінцеві й початкові вершини належать підмножині X_S. Часто буває зручно позначати підграф G_S просто символом $\{X_S\}$; ми будемо надалі використовувати таке позначення, якщо немає небезпеки внесення плутанини.

На рис. 2.4, в показаний породжений підграф графа, наведеного на рис. 2.4, а, що містить тільки вершини x_1, x_2, x_3 і x_4 і дуги, які їх зв'язують.

З'єднуючи наведені вище два визначення, можна сформулювати визначення підграфа. Граф, показаний на рис. 2.4, г, є підграфом графа, наведеного на рис. 2.4, а.

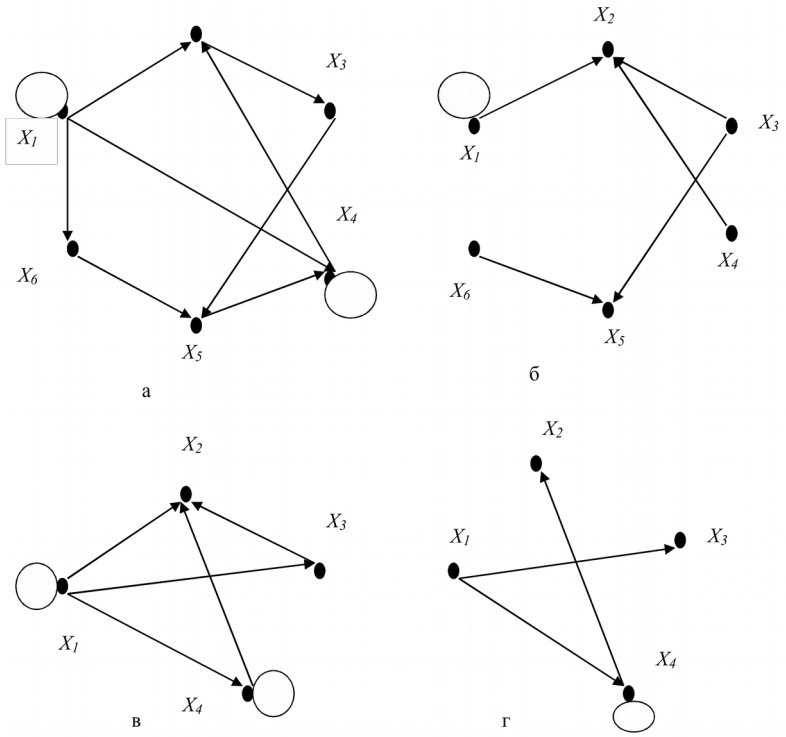


Рис. 2.4 – Граф

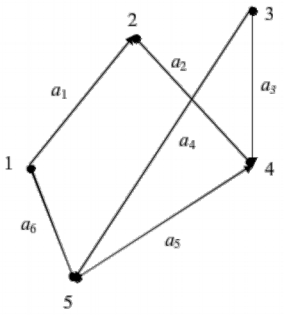
Розглянемо граф, вершини якого представляють співробітників деякої установи, а дуги – лінії зв'язку між співробітниками. Тоді граф, що представляє тільки найбільш важливі канали зв'язку даної установи, є остовим підграфом; граф, що докладно представляє лінії зв'язку тільки якоїсь частини цієї установи (наприклад, відділення), є породженим підграфом, а граф, що представляє тільки важливі лінії зв'язку в межах відділення, є підграфом.

2.3. Матричні представлення графів, операції над графами

1. Для алгебраїчного задавання графів зручно використовувати такі матриці.

Матриця суміжності

Нехай задано граф G (рис. 2.5, а), та його матриця суміжності (рис. 2.5, б), яка позначається через $A = [a_{ij}]$ і визначається в такий спосіб: $a_{ij} = 1$, якщо в G існує дуга (x_i, x_j), і $a_{ij} = 0$, якщо в G немає дуги (x_i, x_j).



$A=$

	X_1	X_2	X_3	X_4	X_5
X_1	0	1	0	0	1
X_2	0	0	0	1	0
X_3	0	0	0	1	1
X_4	0	1	0	0	0
X_5	1	0	0	1	0

а б

Рис. 2.5 – Граф G та його матриця суміжності

Матриця суміжності повністю визначає структуру графа. Наприклад, сума всіх елементів рядка x_i матриці дає напівступінь результату вершини x_i, а сума елементів стовпця x_i – напівступінь заходу вершини x_i. Безліч колонок, що мають 1 у рядку x_i, є безліччю $\Gamma(x_i)$, а безліч рядків, які мають 1 у стовпці x_i, збігається з безліччю $\Gamma^{-1}(x_i)$.

Зведемо матрицю суміжності у квадрат. Нехай елемент $a_{ik}^{(2)}$ матриці A^2 визначається за формулою

$$\grave{a}_{ik}^{(2)} = \sum_{j=1}^{n} a_{ij} a_{jk} . \qquad (2.4)$$

Доданок у рівнянні (2.4) дорівнює 1 тоді й тільки тоді, коли два числа a_{ij} і a_{ik} дорівнюють 1, в іншому випадку він дорівнює 0.

З рівності $a_{ij} = a_{ik} = 1$ випливає існування шляху із двох дуг (ребер), які ідуть із x_i в x_k. Аналогічно, якщо $a_{ik}^{(\delta)}$ є елементом матриці A_p, то $a_{ik}^{(\delta)}$ дорівнює кількості шляхів, які складаються з p дуг (ребер) та ідуть від x_i до x_k.

Матриця інциденцій

Нехай задано граф G з n вершинами й m дугами. Матрицю інциденцій графа G позначимо через $B = [b_{ij}]$, вона і є матрицею розмірності $n * m$, обумовленою в такий спосіб:

$b_{ij} = 1$, якщо x_i є початковою вершиною дуги a_j;

$b_{ij} = -1$, якщо x_i є кінцевою вершиною дуги a_j;

$b_{ij} = 0$, якщо x_i не є кінцевою вершиною дуги a_j або якщо a_j є петлею.

Для графа, наведеного на рис. 2.5, матриця інциденцій має вигляд

$$B = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{array} \begin{array}{c} \begin{array}{cccccccc} a_1 & a_2 & a_2 & a_3 & a_4 & a_5 & a_6 & a_6 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & -1 & -1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 1 & -1 \end{pmatrix} \end{array}.$$

Оскільки кожна дуга інцидентна двом різним вершинам, за винятком того випадку, коли дуга утворить петлю, то кожний стовпець або містить один елемент, що дорівнює 1, і один, що дорівнює –1, або всі елементи стовпця дорівнюють 0. Якщо граф G є неорієнтованим, то матриця інциденцій визначається так само, як і вище, за винятком того, що всі елементи, що дорівнюють –1, заміняються на +1.

Досяжність і зв'язність. Матриці досяжності і контрдосяжності

Матриця досяжності $R = |\mathrm{r_{ij}}|$ визначається в такий спосіб:

$$r_{ij} = \begin{cases} 1, \text{ якщо вершина } x_j \text{ зв'язана з } x_i, \\ 0 \text{ в іншому випадку.} \end{cases}$$

Безліч вершин $R(x_i)$ графа G, досяжних із заданої вершини x_i, складається з таких елементів x_j, при яких (i, j)–й елемент у матриці досяжності дорівнює 1. Очевидно, що всі діагональні елементи в матриці R дорівнюють 1, оскільки кожна вершина досяжна з неї самої за допомогою шляху довжини 0.

Оскільки $\Gamma(x_i)$ є безліччю таких вершин $\mathrm{x_j}$, які досяжні з x_i з використанням шляхів довжини 1 (тобто $\Gamma(x_i)$ – така безліч вершин, для яких у графі існують дуги (x_i, x_j)), і оскільки $\Gamma(x_j)$ є безліччю вершин, досяжних з x_j за допомогою шляхів довжини 1, то безліч $\Gamma(\Gamma(x_i)) = \Gamma^2(x_i)$ складається з вершин, досяжних з x_i з використанням шляхів довжини 2. Аналогічно $\Gamma^p(x_i)$ є безліччю вершин, які досяжні з x_i за допомогою шляхів довжини p.

Оскільки будь–яка вершина графа G, що досяжна з x_i , повинна бути досяжна з використанням шляху (або шляхів) довжини 0 або 1, або 2, ..., або p (з деяким кінцевим, але, можливо, досить більшим значенням p), то безліч вершин, досяжних з x_i, можна подати як

$$R(x_i) = \{x_i\} \cup \Gamma(x_i) \cup \Gamma^2(x_i\} \cup \ldots \cup \Gamma_{\mathrm{p}}(x_i). \qquad (2.5)$$

Таким чином, безліч $R(x_i)$ може бути отримана послідовним виконанням (ліворуч-праворуч) операцій об'єднання в співвідношенні (2.5) доти, поки «поточна» безліч не перестане збільшуватися за розміром при черговій операції об'єднання. Із цього моменту наступні операції не будуть

давати нових членів безлічі й, таким чином, буде утворена досяжна безліч $R(x_i)$. Кількість об'єднань, що потрібно виконати, залежить від графа, але, мабуть, що число p менше кількості вершин у графі.

Матрицю досяжності можна побудувати так. Знаходимо досяжні безлічі $R(x_i)$ для всіх вершин $x_i \in X$ способом, наведеним вище. Покладемо r_{ij} =1, якщо $x_j \in R(x_i)$, і $r_{ij} = 0$ в іншому випадку. Отримана в такий спосіб матриця R є матрицею досяжності.

Матриця контрдосяжності $Q = |g_{ij}|$ визначається в такий спосіб:

$$q_{ij} = \begin{cases} 1, \text{ якщо від вершини } x_j, \text{ можливо, досягає вершини } x_i, \\ 0 \text{ в іншому випадку.} \end{cases}$$

Контрдосяжністю безлічі $Q(x_i)$ графа G є безліч таких вершин, що з будь–якої вершини цієї безлічі можна досягнути вершини x_i. Аналогічно побудові досяжної безлічі $R(x_i)$ можна сформувати безліч $Q(x_i)$:

$$Q(x_i) = \{x_i\} \cup \Gamma^{-1}(x_i) \cup \Gamma^{-2}(x_i\} \cup \ldots \cup \Gamma^{-P}(x_i), \qquad (2.6)$$

де $\Gamma^{-2}(x_i) = \Gamma^{-1}(\Gamma^{-1}(x_i))$.

Операції виконуються ліворуч-праворуч доти, поки чергова операція об'єднання не перестане змінювати «поточну» безліч $Q(x_i)$.

З визначень очевидно, що стовпець x_i матриці Q (у якому $q_{ij} = 1$ $x_i \in Q(x_i)$ і $q_{ij} = 0$ в іншому випадку) збігається з рядком x_i матриці R, тобто $Q = R^t$, де R^t – матриця, транспонована до матриці досяжності R.

Відповідний приклад наведено на рис. 2.6.

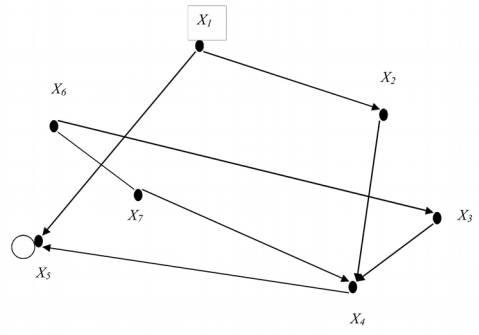


Рис. 2.6 – Граф G

Завдання для закріплення наведеного матеріалу: знайти матриці досяжності і зворотних досяжностей для графа G, наведеного на рис. 2.6. Матриця суміжності графа

$$A = \begin{array}{c|ccccccc} & X_1 & X_2 & X_3 & X_4 & X_5 & X_6 & X_7 \\ \hline X_1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ X_2 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ X_3 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ X_4 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ X_5 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ X_6 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ X_7 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{array}$$

.

$R(x_1) = \{x_1\} \cup \{x_2, x_5\} \cup \{x_2,, x_4, x_5\} \cup \{x_2, x_4, x_5\} = \{x_1, x_2, x_4, x_5\}$.

Операції над графами

Видалення вершини або ребра, а також перехід до підграфа – це операції, за допомогою яких можна з наявного графа одержувати інші графи з меншою кількістю елементів. Відомі також операції, що дозволяють, навпаки, одержувати з наявних графів «більші» графи. Такою є, наприклад, операція додавання ребра: якщо вершини u і v графа G несуміжні, то можна визначити граф $G+e$, де $e = uv$. Його отримують із графа G додаванням ребра e.

Однією з найбільш важливих є операція об'єднання. Граф H називається об'єднанням (або накладенням) графів F і G, якщо $VH = VF \cup VG$ і $EH = EF \cup EG$ (рис. 2.7). У цій ситуації пишуть $H = F \cup G$. Об'єднання $F \cup G$ називається диз'юнктивним, якщо $VF = VG$. Аналогічно визначаються об'єднання й диз'юнктивне об'єднання будь–якої безлічі графів, причому в останньому випадку ніякі два з поєднуваних графів не повинні мати загальних вершин.

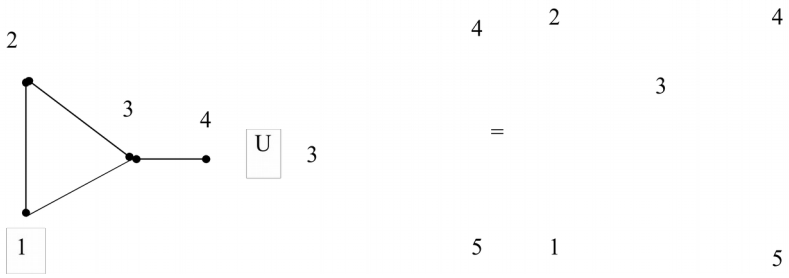


Рис. 2.7 – Операції над графами

Нехай $G_i = (V_i, E_i)$ $(I = 1,2)$ – два графи. Добутком $G_1x_2=G$ називається граф, для якого $VG= V_1x$ 2–декартовий добуток множин вершин вихідних графів, а EG визначається в такий спосіб: вершини (u_1, u_2) і (v_1, v_2) суміжні в графі G тоді й тільки тоді, коли або $u_1= v_1$, а u_2 і v_2 суміжні в G_2, або $u_2 = v_2$, а u_1 і v_1 суміжні в G_1 (рис. 2.8). Розглянемо також операцію стягування ребра. Стягування ребра iw означає ототожнення суміжних вершин i та w. На рис. 2.9 показані граф G і граф, отриманий з G стягуванням ребра (1, 2).

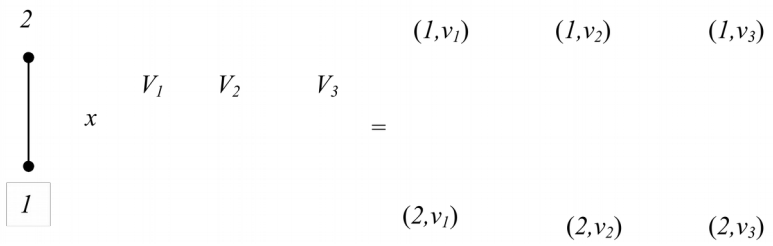


Рис. 2.8 – Добуток G_1x_2

Рис. 2.9 – Стягування ребра

Граф G називається стягуваним до графа *H*, якщо *H* отримують із *G* у результаті деякої послідовності стягувань ребер. Очевидно, що будь–який непустий зв'язний граф, відмінний від K_1, стягаємо до K_2. Але вже не будь–який зв'язний граф стягується до графа K_3. Наприклад, простий ланцюг P_n не стягується до K_3. Природно виникає параметр $\eta(G)$– максимум порядків повних графів, до яких стягується граф *G*. Параметр ((*G*) називається числом Хадвігера графа *G*. Це число пов'язане з проблемою чотирьох фарб.

У певному розумінні двоїстою операцію стягування ребра є операція розщеплення вершини. Нехай *v* – одна з вершин графа *G*. Розіб'ємо оточення довільно на дві частини *M* і *N* і виконаємо таке перетворення графа *G*: видалимо вершину *v* разом з інцидентними їй ребрами, додамо нові вершини *u* і *w* і з'єднуюче їхнє ребро *uw*, вершину *u* з'єднаємо ребром з кожною вершиною з безлічі *M*, а вершину v – з кожною вершиною з безлічі *N*. Отриманий у результаті граф позначимо символом $G^{\wedge}$. Говорять, що $G^{\wedge}$ отримують із *G* розщепленням вершини v (рис. 2.10).

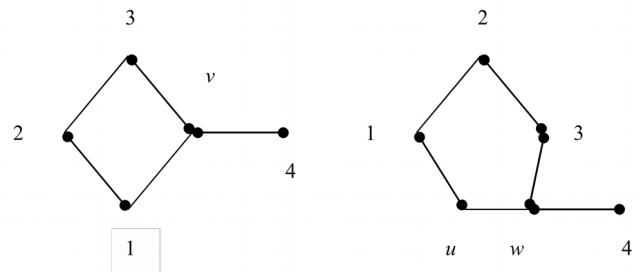


Рис. 2.10 – Розщеплення вершини графа

Ізоморфізм графів

Розглянемо таку важливу задачу. Якщо в нас є два подання мережі у вигляді графа, то як довідатися, чи не описують вони ту саму мережу.

Очевидно, спочатку потрібно визначити, що мається на увазі, коли ми говоримо, що «дві мережі однакові». Два графи G і G' *ізоморфні,* якщо існує взаємнооднозначна відповідність між вершинами G і G', така, що дві вершини u і v суміжні в G тоді й тільки тоді, коли в G' суміжні відповідні їм вершини. Точніше, $G=(V, E)$ - ізоморфний $G' = (V', E')$, це записується у вигляді $G \approx G'$, якщо існує взаємнооднозначна функція h з V у V', що має таку властивість, що $(u, v) \in E$ тоді й тільки тоді, коли $(h(\mathrm{u}), h(v)) \in T \backslash E'$.

Проблема визначення, чи є дві мережі ізоморфними, зненацька виявляється важкою. Для довільних мереж всі відомі алгоритми, що гарантують правильну відповідь, – експонентні. Фактично навіть для невеликих мереж це завдання вирішити нелегко. Всі три мережі на рис. 2.11 ізоморфні одна одній.

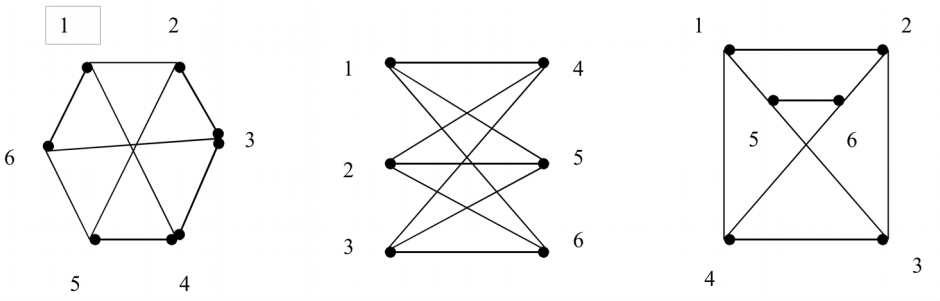


Рис. 2.11 – Ізоморфні графи

При перенумерації вершин у графі G міняються місцями відповідні рядки й стовпці, оскільки з n вершин можливо n! перестановок та існує $n!$ графів, ізоморфних даному графу G. Для встановлення ізоморфізму графів G і G' нам необхідно знайти таку перенумерацію вершин в одному з графів, при якій матриці суміжності обох графів будуть однаковими. Зрозуміло, що експонентне число можливих рішень істотно ускладнюють можливості вирішення даного завдання. На сьогоднішній день навіть невідомо, до якого класу завдань належить задача визначення ізоморфізму графів. Ніхто не зміг довести її приналежність до класу *NP*, але й ніхто не зміг побудувати поліноміальні алгоритми її розв'язання. Виключення становить задача визначення ізоморфізму орієнтованих графів, для якої отримані поліноміальні алгоритми її розв'язання. Із завданням ізоморфізму графів тісно пов'язане поняття інваріанта графа. Інваріантом графа G називають параметр, що має те саме значення для всіх графів ізоморфних G. Серед найочевидніших відзначимо такі: кількість вершин, кількість ребер, кількість компонентів, послідовність ступенів вершин, тобто список ступенів всіх вершин у порядку зменшення значень. Наприклад, матриця

суміжності не інваріант графа, оскільки при перенумерації вершин графа вона зазнає перестановки рядків і стовпців. Будь–яка функція f від елементів a_{ij} незмінювана ні при яких перестановках рядків і стовпців матриці суміжності $A(a_{ij})$ і є інваріантом графа G. До числа таких функцій належать сума всіх елементів, неупорядкований набір сум всіх елементів кожного рядка або стовпця, визначник матриці, її характеристичний багаточлен та ін. Інваріант f називається повним, якщо для будь–яких графів G і G' з рівності інваріантів $f(G) = f(G')$ впливає, що графи G і G' ізоморфні. При вивченні спектральних властивостей графів було встановлено, що повним інваріантом є міні- й макси–код матриці суміжності $A(a_{ij})$ графів. Через симетричність матриці $A(a_{ij})$ для її задавання достатньо виписати в певному порядку лише ті елементи, які розташовані над головною діагоналлю, тобто задати кортеж $(a_{12}, a_{13}, a_{23}, a_{14}, a_{24}, a_{34}, \ldots, a_{n-1,n})$ довжини $\tilde{N}_2^n$. Число $a_{12}*2_0+ a_{13}*2^1+ a_{23}*2^2+ a_{14}*2^3+ \ ,\ldots,+a_{n-1,n}*2^{\tilde{n}_2^n-1}$, при записі якого у двійковій системі кількість одиниць дорівнює a_{12}, кількість двійок a_{13}, кількість четвірок a_{23} і та ін. назвемо двійковим кодом матриці $A(a_{ij})$. Найменший із цих кодів (при всіх можливих n! нумераціях) називають міні–кодом $\mu_*(G)$, а найбільший макси–кодом $\mu^*(G)$ графа G. За кожним з них і кількістю вершин легко відновлюється одна з матриць суміжностей графа, а отже, і сам граф з точністю до ізоморфізму. Наприклад, розглянемо граф G на рис. 2.12, а.

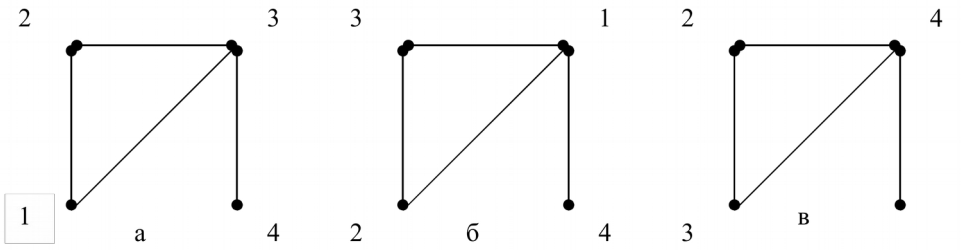


Рис. 2.12 – Графи

Граф на рис. 2.12, а припускає 4!=24 різних нумерації вершин. Через симетрію в структурі графа перестановка одного з одним номерів вершин ступеня не змінює матрицю суміжності й тому різних матриць буде не 24, а 12. Нумерації на рис. 2.12, б відповідає міні-код $\mu_*(G)$=15, а нумерації на рис. 2.12, в макси-код $\mu^*(G)$. Для нумерації, наведеної на рис. 2.12, б, маємо

$$\begin{vmatrix} 0111 \\ 1010 \\ 1100 \\ 1000 \end{vmatrix} = 1*2^0+1*2^1+1*2^2+1*2^3+0*2^4+0*2^5=15.$$

При розв'язанні задачі ізоморфізму графів нам необхідно знайти повний інваріант, що легко обчислюється, але, на жаль, для розв'язання задачі ізоморфізму довільних графів обчислення всі відомі повні інваріанти вимагають експонентного часу.

Слід зазначити, що встановлення факту неізоморфності графів на практиці в більшості випадків здійснюється досить легко. Але встановлення факту ізоморфізму графів є важким, оскільки в цьому випадку потрібно вказати підстановку, що переводить матрицю суміжності одного графа в матрицю суміжності іншого.

Так, якщо розглянути два ізоморфних графи (рис. 2.12, а і б), то підстановка

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 1 & 4 \end{pmatrix}$$

переводить матрицю суміжності графа на рис. 2.12, а у матрицю суміжності графа на рис. 2.12, в.

Група автоморфізмів графа

Однією з важливих характеристик графів є група автоморфізмів графа. Автоморфізмом графа G називається підстановка φ на безлічі вершин графа G, що зберігає відношення суміжності, тобто така, що образи φ(u) і φ(v) вершин u і v є суміжними тоді й тільки тоді, коли суміжні самі вершини u і v. Іншими словами, автоморфізм графа – це ізоморфізм графа на себе. Будь–який граф має щонайменше один автоморфізм – тотожне перетворення e: $VG \to VG$, при якому $e(v) = v$ для будь–якої вершини v. Очевидно, що якщо φ – автоморфізм графа G, то й зворотна підстановка φ^{-1} теж є автоморфізмом, якщо ж підстановки φ і η - обидві суть автоморфізму, то і їхній добуток $\varphi*\eta$ – автоморфізм. Тому правильним є таке твердження: безліч всіх автоморфізмів графа відносно операції множення підстановок є групою.

Довідка. Групою називається безліч Ω елементів довільної природи, у якому:

1) задано правило, відповідно до якого будь–якій парі (a, c) елементів Ω відповідає деякий елемент $a*c$ тієї самої безлічі;

2) групова операція асоціативна, тобто для будь–яких елементів a, c і $к$ з Ω $(a*c)*к = a*(c*к)$;

3) в Ω є нейтральний елемент, тобто такий елемент e, що для будь–якого $a \in$ Ω $a*e = e*a = a;$

4) у кожного елемента a в Ω є симетричний, тобто такий елемент a', що $a*a' = a'*a=e.$

Властивості 1 – 4 називають аксіомами групи. Група автоморфізмів графа G позначається через $AutG$.

Уведемо важливе поняття орбіти групи підстановок. Нехай Γ – довільна група підстановок на безлічі V. Визначимо на безлічі бінарне відношення ~, поклавши $u \sim v$ для u, $v \in V$ тоді й тільки тоді, коли в Γ існує така підстановка s, що $s(u) = v$. Очевидно, що відношення ~ є відношенням еквівалентності й, отже, безліч V розбивається на класи еквівалентних елементів: всі елементи, що входять в один клас, переводяться підстановками із групи Γ один в одного, а елементи з різних класів один в одного не переводяться. Ці класи називаються орбітами групи Г. Розбиття безлічі вершин графа G на орбіти групи $AutG$ – важливе завдання. По суті, застосування до графа автоморфізму означає перенумерацію вершин, причому відношення суміжності повинне зберігатися. Відомо, проблема розпізнавання приналежності двох вершин довільного графа одній орбіті його групи автоморфізмів і проблема ізоморфізму графів еквівалентні в тому розумінні, що будь–який алгоритм, що вирішує одну із цих проблем, може бути ефективно перетворений для вирішення іншої.

2.4. Незалежні безлічі й верхові покриття

Домінуючі безлічі

Нехай дано граф $G = (X, \Gamma)$. Досить часто виникає задача пошуку таких підмножин безлічі вершин X графа G, які мають певну, наперед задану властивість. Наприклад, якою є максимально можлива потужність такої підмножини $S \subseteq X$, для якої породжений підграф (S) є повним? Або яка максимальна потужність підмножини S, такої, що граф (S) – цілком незв'язний? Відповідь на перше запитання дає так зване клікова кількість графа G, а на друге – кількість незалежності. Ще одна задача. Вона полягає в знаходженні мінімально можливої потужності таких підмножин S безлічі X, що будь–яка вершина з $X - S$ досяжна з S за допомогою шляхів одиничної довжини. Розв’язання цієї задачі дається так званим числом домінування графа G.

Ці числа й пов'язані з ними підмножини вершин описують важливі структурні властивості графа й мають різноманітні безпосередні додатки при веденні проектного планування дослідницьких робіт, у кластерному аналізі й чисельних методах, паралельних обчисленнях на ЕОМ, при розміщенні підприємств обслуговування, а також джерел і споживачів в енергосистемах.

Розглянемо неорієнтований граф $G = (X, \Gamma)$. Незалежна безліч вершин (відома також як внутрішньо стійка безліч [11]) є безліччю вершин графа G, такою, що будь–які дві вершини в ній не суміжні, тобто ніяка пара вершин не з'єднана ребром.

Незалежна безліч називається максимальною, коли немає іншої незалежної безлічі, у яку вона б входила. Слід також зазначити, що кількість елементів (вершин) у різних максимальних безлічах, як випливає з наведеного вище прикладу, не обов'язково однакова.

Якщо Q є сімейством всіх незалежних безлічей графа G, то число

$$\alpha[G] = \max_{s \in Q} |S|$$

називається числом незалежності графа *G*, а безліч *S**, на якій цей максимум досягається, називається найбільшою незалежною безліччю.

Максимальні повні підграфи (кліки)

Поняття, протилежне максимальній незалежній безлічі, є максимальним повним підграфом. Таким чином, максимальний повний підграф (клік) графа *G* є породженим підграфом, побудованим на підмножині *S* вершин графа і є повним і максимальним у тому розумінні, що будь–який інший підграф графа *G*, побудований на безлічі вершин H, що містить *S*, тобто $H \supset S$, не є повним. Отже, на противагу максимальній незалежній безлічі, у якій не можуть зустрітися дві суміжні вершини, у кліці всі вершини попарно суміжні. Також очевидно, що максимальна незалежна безліч графа *G* відповідає кліці графа $G^{\sim}$ і навпаки, де $G^{\sim}$ – доповнення графа *G*.

Цілком очевидно також, що поняття кліки для неорієнтованого графа подібно поняттю кліка для графа. Клік у дійсності можна розглядати як такий сильний компонент, у якому досяжність обмежена шляхами одиничної довжини.

Аналогічно тому, як було визначена кількість незалежності графа, ми можемо визначити клікове число графа (відоме також як щільність).

Це – максимальна кількість вершин у кліках даного графа. Тоді, образно кажучі, у «щільного» графа клікове число буде, імовірно, більше, а число незалежності менше, у той час як у «розрідженого» графа, цілком імовірно, буде мати місце протилежне співвідношення між цими числами.

2.5. Планарність і розфарбування графів

Планарні графи

Кажуть, граф *G* укладається на поверхні *S,* якщо його можна зобразити на поверхні таким чином, що його ребра перетинаються тільки біля своїх кінцевих вершин. Граф називається планарним, якщо його можна укласти на площині (рис. 2.13).

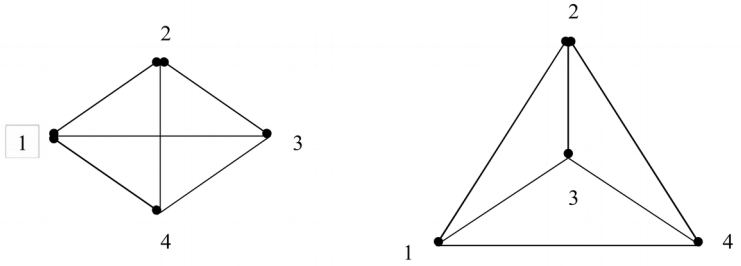


Рис. 2.13 – Планарний граф

Справедливі такі твердження:

для кожного простого планарного графа (що не містить мульти-ребер) існує планарне укладання, у якій всі ребра можна зобразити у вигляді відрізків прямих ліній;

граф укладається на площині тоді й тільки тоді, коли він укладається на сфері (тобто поняття укладання графа на сфері й площині еквівалентні).

Два основних непланарних графи називають графами Куратовського. Один з них K_5 – повний граф з п'ятьма вершинами (рис. 2.14, а), а інший двочастковий граф $K_{3,3}$ з трьома вершинами в кожній частці (рис. 2.14, б).

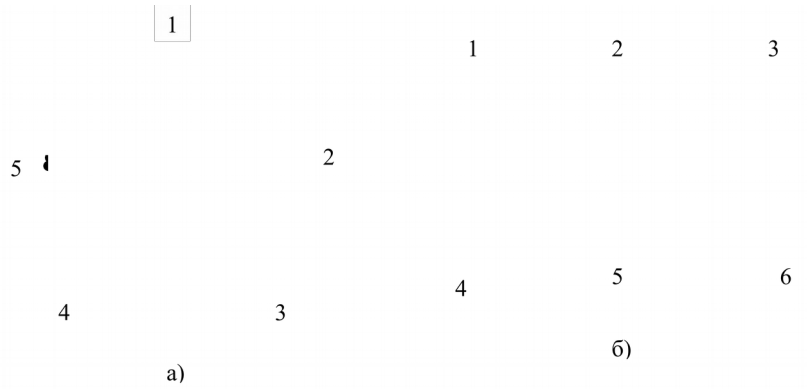


Рис. 2.14 – Графи Куратовського

Формула Ейлера

Укладання графа на площині ділять його на області, які в графі утворять базис підпростору циклів графа G. Із цього факту, як показав Ейлер, випливає справедливість формули

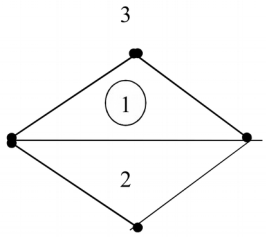
$$n - m + r = 2,$$

де n – кількість вершин у графі;

m – кількість ребер у графі;

r – кількість областей на площині.

*Приклад реалізації формули Ейлера наведено рис. 2.15***.**



$4 - 5 + 3 = 2$

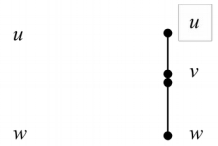
Рис. 2.15 – Плоский граф

Якщо планарний граф G має m – ребер і $n \geq 3$, то $m \leq 3n - 6$. У планарному є принаймні одна вершина ступеня не більше 5.

Гомеоморфні перетворення графів

1. Операція стягування або злиття ребер.

2. Операція включення вершини.



Операції 1, 2 називаються гомеоморфними перетвореннями графа. Два графи називаються гомеоморфними, якщо вони ізоморфні або можуть стати ізоморфними в результаті гомеоморфних перетворень. Очевидно, що якщо граф планарний, то будь–який граф, гомеоморфний йому, є теж планарним.

Куратовським була доведена наступна теорема.

Теорема 2.1. Граф планарний тоді й тільки тоді, коли він не містить підграфа, гомеоморфного графам K_5 або $K_{3,3}$.

Цей самий результат був отриманий Понтрягіним, тому дану теорему називають теоремою Понтрягіна – Куратовського.

Приклад 2.2 до теореми 2.1.

Граф на рис. 2.16 називається графом Петерсона, він не містить підграфа, ізоморфного графам K_5 і $K_{3,3}$, але він не планарний, оскільки виражений після стягування виділених ребер у граф K_5.

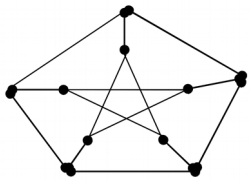


Рис. 2.16 – Граф Петерсона

Розфарбування графів

Граф G називається r–хроматичним, якщо його вершини можуть бути розфарбовані з використанням r–кольорів (фарб), так що не знайдеться двох суміжних вершин, пофарбованих в один колір. Найменше число r, таке, що граф G є r–хроматичним, називається хроматичним числом графа. Позначимо його γ(G). Завдання знаходження хроматичного числа графа називається завданням розфарбування графа. Відповідному хроматичному числу розфарбування вершин графа розбиває безліч вершин на r підмножин, кожна з яких містить вершини одного кольору. Ці безлічі називаються незалежними, оскільки в межах однієї безлічі немає суміжних вершин.

Оцінки хроматичних чисел графів

Кліковим числом $P(G)$ графа G називається найбільша кількість вершин у повному породженому підграфі графа G. Оскільки принаймні $P(G)$ кольорів потрібно для розфарбування відповідної кліки графа G, то $P(G)$ є нижньою оцінкою хроматичного числа, тобто справедлива нерівність $\gamma(G) \geq P(G)$.

Число незалежності графа $\alpha(G)$ дорівнює потужності найбільшої безлічі попарно несуміжних вершин графа і відповідає потужності найбільшої кількості вершин у G, які можуть бути пофарбовані в один колір, отже, справедлива нерівність

$$\gamma(G) \geq \lceil n/\alpha(G) \rceil,$$

де n – кількість вершин графа G;

$\lceil X \rceil$ – найбільше ціле число, що не перевищує X.

Відомі і верхні оцінки хроматичного числа, але їх використати в алгоритмах розфарбовування неможливо.

Розфарбування планарних графів

З розфарбуванням планарних графів тісно пов'язане завдання розфарбування карти, що формулюється так: чи вистачить чотирьох фарб для розфарбування довільної карти, на якій будь-які дві сусідні країни пофарбовані в різні кольори? У 1879 році британський математик Келі висловив гіпотезу, що будь-яка карта розфарбовується в чотири кольори. Якщо кожній країні поставити у відповідність вершини деякого графа G і з'єднати кожну вершину з тими, з якими межує відповідна країна, то ми

одержимо планарний (плоский) граф, правильне вершинне розфарбування якого визначає розфарбування карти. Гіпотеза чотирьох фарб залишилася математично не доведеною, але проведено машинний доказ цієї теореми групою математиків на чолі з Аппелем. Повторити цей експеримент ніхто не зміг.

Зведення задачі про розфарбування графів до ЗНП

Оскільки при будь–якому припустимому розфарбуванні графа G безліч вершин фарбуються в один колір повинна бути незалежною безліччю, то будь-яке припустиме розфарбування можна інтерпретувати як розбиття всіх вершин графа G на незалежні безлічі. Якщо кожну незалежну безліч розширити до максимального шляхом додавання до неї інших вершин, то розфарбування графа може бути витлумачено як покриття вершин графа G максимальними незалежними безлічами. При цьому деякі вершини можуть належати більш ніж одній незалежній безлічі. Це говорить про можливість існування різних припустимих розфарбувань, використовуючи одну й ту саму кількість кольорів.

Так, нехай побудовані все максимально незалежні безлічі $\{t\}$ графа G і задана матриця $M = [m_{ij}]$ розмірності $n*t$, у якій $m_{ij} = 1$, якщо вершина i належить j–й максимальній незалежній безлічі, і $m_{ij} = 0$ в іншому випадку. Якщо тепер кожній незалежній безлічі зіставити одиничну вартість, то задача розфарбування зведеться до знаходження найменшої кількості стовпців у матриці M, що покриває всі її рядки. Кожний стовпець із розв'язання даної ЗНП відповідає певному кольору, який може бути використаний для розфарбування вершин максимальної незалежної безлічі, представленої даним стовпцем.

Формулювання задачі розфарбування як ЗНП переважніше, ніж безпосередня її постановка, як задачі 0, 1–програмування, оскільки в цьому випадку вдається розв'язувати ЗНП значно більшої розмірності.

Нехай задано граф (рис. 2.17). Він має незалежні множини: {3, 5}; {2, 4}; {2, 5}; {1}; матрицю М

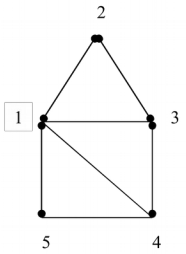


Рис. 2.17 – Граф

	3, 5	2, 4	2, 5	1
1	0	0	0	1

2	0	1	1	0
3	1	0	0	0
4	0	1	0	0
5	1	0	1	0

2.6. Цикли в графах

Ейлерові цикли

Початок теорії графів пов'язують із завданням про кенігсберзькі мости. Це завдання полягає от у чому. Сім мостів міста Кенігсберга (нині Калінінград) були розташовані на ріці Прегель. Запитується, чи можна, вийшовши з будинку, повернутися назад, пройшовши в точності один раз по кожному мосту.

Зіставимо плану міста - граф *G* (рис. 2.18), вершини якого відповідають чотирьом поділеним рікою ділянкам суші А, В, С, Д, а ребра – мостам. Мовою теорії графів це завдання формулюється так: чи є в графі *G* цикл, що містить всі ребра графа. У 1736 році Ейлер довів нерозв'язність задачі про кенігсберзькі мости й вирішив таку загальну проблему теорії графів: при яких умовах зв'язний граф містить цикл, що проходить через кожне ребро графа. Зв'язний граф, у якому є ейлерів цикл, називається ейлеровим графом.

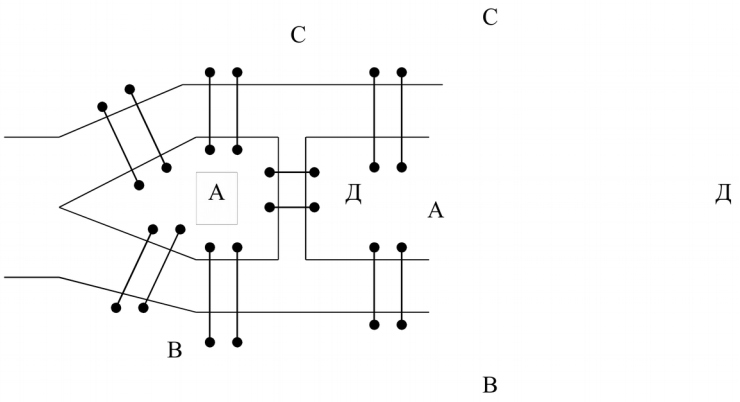


Рис. 2.18 – Граф схеми мостів міста

Теорема 2.2. Зв'язний граф є ейлеровим тоді й тільки тоді, коли ступені всіх його вершин парні.

Нехай *G* – ейлерів граф (рис. 2.19). Ейлерів цикл цього графа, проходячи через кожну його вершину, входить у цю вершину по одному ребру, а виходить по іншому. Це означає, що кожна вершина інцидентна парній кількості ребер ейлеревого циклу, а оскільки такий цикл містить ребра графа G, то звідси випливає парність ступенів його вершин.

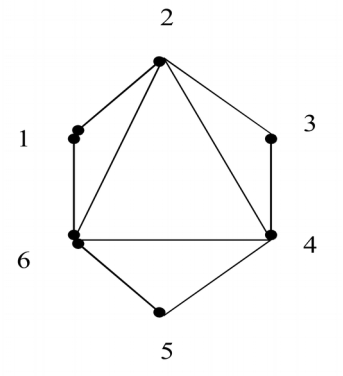


Рис. 2.19 – Ейлерів граф

Наприклад, граф на рис. 2.19 є ейлеровим графом, оскільки він містить цикл (1, 2, 3, 4, 6, 4, 2, 6, 1). У цьому графі є й інші ейлерові цикли.

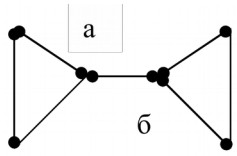
Важливим наслідком, що випливає з теореми 2.2, є справедливість такого твердження: якщо зв'язний граф містить рівно K вершин, то мінімальна кількість покриваючих його ребра непересічних ланцюгів дорівнює $K/2$. Ланцюг, що містить всі ребра графа, називається ейлеровим. Ейлерів граф, як правило, містить декілька ейлерових циклів. Знаючи один цикл, можна одержати новий за допомогою такого простого прийому. Нехай C - вихідний ейлерів цикл, і вершина v проходиться в цьому циклі більше одного разу. Розглянемо частину циклу C (коцикл), який складається з ребер і вершин прохідних між $к$ – м і l – м відвідуваннями вершини v ($к < l$), позначимо цей підцикл C_1. Цикл задає деякий порядок проходження ребер C_1. Якщо порядок проходження ребер C_1 змінити на зворотний, а на інших ребрах залишити його таким, яким він був у C, то, очевидно, одержимо новий ейлерів цикл C'. Наприклад, у графі, зображеному на рис. 2.19, можна одержати новий цикл C' = (1, 2, 6, 5, 4, 3, 2, 4, 6, 1), покладаючи C = (1, 2, 3, 4, 6, 4, 2, 6, 1) і C_1 = (2, 3, 4, 5, 6, 2). У цьому прикладі цикл складений з ребер, що проходять у C між першим і другим відвідуванням вершини 2. Виникає запитання, а як знайти хоча б один цикл в ейлеровому графі, тобто як занумерувати ребра графа числами 1, 2, ... m для того, щоб номер, привласнений ребру, указував, яким це ребро проходить в ейлеровому циклі?

Процедура нумерації ребер

1. Починаючи з довільної вершини u, привласнюємо довільному ребру (u, v) номер 1, потім викреслюємо ребро (u, v) і переходимо у вершину v.

2. Нехай w – вершина, у яку ми прийшли в результаті виконання попереднього кроку, і K – номер, привласнений деякому ребру на цьому кроці. Вибираємо будь–яке ребро інцедентної вершині w, причому міст вибираємо в тому випадку, коли немає інших можливостей (мостом називається ребро графа, якщо його видалення збільшує кількість компонентів графа).

Ребро (а, б) утворює міст.



Привласнюємо ребру номер K+1. Цей процес, називаний алгоритмом Флері, закінчується, коли всі ребра графа викреслені, тобто пронумеровані. Зазначимо, що ейлерові графи досить рідкі. Доведено таку теорему: майже немає ейлерових графів. Але неейлерові графи легко перетворити в ейлерові, доповнюючи їх фіктивними вершинами й ребрами, задовольняючи теорему 2.2.

Гамільтонові цикли

Гамільтоновим циклом у графі G називається цикл, що містить всі вершини графа. Граф називається гамільтоновим, якщо він має гамільтонів цикл. На відміну від ейлерових циклів, визначити, чи існує в графі гамільтонів цикл, надзвичайно важко. У теорії графів є кілька теорем, що встановлюють гамільтоновість графів, але всі вони мають більш теоретичне значення. Якщо ребрам графа привласнені деякі ваги C_{ij}, і граф G містить декілька гамільтонових циклів і гамільтонових шляхів, то великий інтерес представляє завдання визначення найкоротших гамільтонових циклів і шляхів. Дана задача відома як задача про комівояжера, складність її розв'язання обумовлена тим, що кількість розв'язків пропорційно $n!$.

3. Зв'язок між ейлеровими й гамільтоновими циклами.

Подвійність між ейлеровими й гамільтоновими циклами (заміна вершини на ребро й навпаки) призводить до тісного зв'язку між цими двома поняттями в застосуванні до неорієнтованого графа G і відповідного йому реберного графа, для якого дамо визначення.

Реберний граф G_c має стільки ж вершин, скільки ребер у графа G. Ребро між двома вершинами графа G_c існує тоді й тільки тоді, коли ребра графа G, що відповідають цим двом вершинам, суміжні (тобто інцидентні з однією й тією самою вершиною графа G). Справедливі такі твердження:

якщо G має ейлерів цикл, то G_c має як ейлерів, так і гамільтонів цикли;

якщо G має гамільтонів цикл, то G_c також має гамільтонів цикл.

Зворотні твердження, на жаль, не є правильними.

Цикломатичне число й фундаментальні цикли

Нехай G – неорієнтований S-граф з n вершинами, m ребрами (S-граф – це граф, що має мультиребра, при цьому кількість паралельних ребер не перевищує S). Визначимо кількість $\rho(G)$ як $\rho(G)$ = n – p у кожному з p-зв'язних компонентів графа G.

Число $\nu(G) = m - \rho(G) = m - n + p$ називається цикломатичним числом графа G , а $\rho(G)$ – коцикломатичним числом.

Наприклад, граф складається з одного зв'язного компонента. Ост Т графа показаний більш жирним контуром (рис. 2.20).

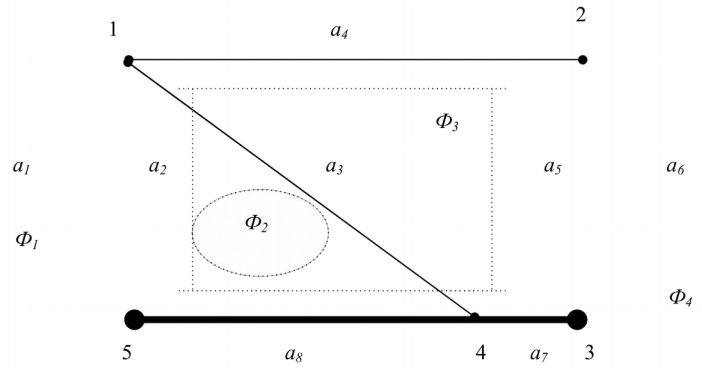


Рис. 2.20 – Неорієнтований S – граф

Додавання будь–якого ребра (i, j) з G, що не належить T, призводить до утворення одного простого циклу. Цикломатичне число графа дорівнює $m - n + p = 8 - 5 + 1 = 4$. Цикли Φ_1, Φ_2 ,Φ_3 ,Φ_4 незалежні, тому що кожний з них має принаймні одне ребро, що не належить ніякому іншому циклу. У загальному випадку $\nu(G)$–цикли, одержувані додаванням якогось ребра з G, що не належить Т, називають фундаментальною безліччю циклів, а самі цикли фундаментальними. Будь-який інший цикл у графі може бути виражений у вигляді лінійної комбінації з безлічі фундаментальних циклів. Для цього представимо кожний фундаментальний цикл m–мірним вектором, у якому j-та компонента дорівнює 1 або 0 залежно від того, належить j-те ребро даному циклу. Тоді безліч циклів у графі може бути задано матрицею циклів

$\Phi =$

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
Φ_1	1	1	0	0	0	0	0	0
Φ_2	0	1	1	0	0	0	0	1
Φ_3	0	1	0	1	1	0	1	1
Φ_4	0	0	0	0	1	1	0	0
Φ_5	1	0	0	1	1	0	1	1

.

Матриця циклів Φ складається з $\nu(G)$ рядків і m-стовпців, у якій $\varphi_{ij} = 1$, якщо ребро a_j належить циклу Φ_i, і дорівнює 0 в іншому випадку.

Використовуючи символ $\oplus$ додавання по модулю 2, будь-який цикл $\Phi_к$ можна представити як суму по модулю 2 фундаментальних циклів.

Наприклад, цикл $\Phi_5 = (a_1 ,a_4 ,a_5 ,a_7 ,a_8)$ може бути представлений як $\Phi_5 = \Phi_1 \oplus \Phi_3 = 11000000 \oplus 01011011 = 10011011$.

Слід зазначити, що деякі комбінації фундаментальних циклів можуть не давати цикл, а самі фундаментальні цикли визначені не однозначно й залежать від випадково обраного остова Т.

2.7. Центри графів

Розміщення центрів графа

У практичній діяльності постійно виникають задачі найкращого розміщення встаткування в мережах або графах. До таких задач належить задача розміщення служб відновлення мережі при виникненні відмов у мережі. У цьому випадку критерій оптимальності може полягати в мінімізації відстані або часу проїзду від пункту обслуговування до місця виникнення відмови. У більш загальній задачі потрібно розмістити кілька таких пунктів обслуговування. При цьому найвіддаленіша вершина графа повинна перебувати принаймні хоча б від одного пункту обслуговування на мінімальній відстані. Задачі такого типу називаються мінімаксними задачами розміщення. Отримані при розв'язанні цих задач місця розміщення пунктів обслуговування називаються центрами графа. У деяких задачах розміщення потрібно мінімізувати суму всіх відстаней від вершин до центра обслуговування, задачі такого типу належать до мінісумних задач розміщення. Місця розміщення пунктів обслуговування, отримані в результаті розв'язання мінісумної задачі, називаються медіанами графа.

Числа поділу в графах

Для будь-якої вершини X_i у графа $G(X, E)$ нехай $R^0{}_\lambda(x_i)$ є безліччю тих вершин $\{X_j\}$ графа G, яких можна досягнути з вершини X_i за допомогою шляхів зі зваженими довжинами $v_j d(X_i, X_j)$, що не перевищує величини λ. Через $R^t{}_\lambda(x_i)$ позначимо безліч тих вершин $\{X_j\}$, з яких X_i може бути досягнута з використанням шляхів, що мають зважені довжини $v_j d(X_j, X_i) < \lambda$. У такий спосіб

$R^0{}_\lambda(x_i)=\{ X_j \mid v_j d(X_i, X_j) \le \lambda;\ X_j \in X\}$,

$R^t{}_\lambda(x_i)=\{ X_j \mid v_j d(X_j, X_i) \le \lambda;\ X_j \in X\}$.

Для кожної вершини визначимо такі два числа:

$S_0(X_i)=\max [v_j d(X_i, X_j)]$,

$S_t(X_i)= \max [v_j d(X_j, X_i)]$.

Числа $S_0(X_i)$, $S_t(X_i)$ відповідно називаються числом зовнішнього поділу й числом внутрішнього поділу вершини X_i. Слід зазначити, що $S_0(X_i)$ є найбільшим числом у рядку X_i матриці $D'(G)$, отриманої в результаті множення кожного стовпця j матриці відстаней $D(G)=\| d(X_i, X_j)\|$ на v_j, і $S_t(X_i)$ є найбільшим числом у стовпці X_i матриці $D''(G)$, отриманої після множення кожного рядка матриці відстаней $D(G)$ на v_j. Розглянемо як приклад орієнтований граф на рис. 2.21.

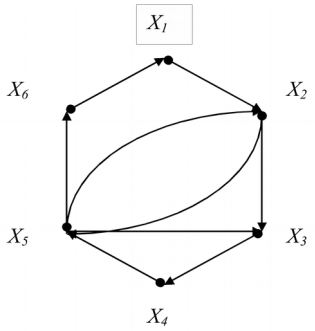


Рис. 2.21 – Орієнтований граф

Матриця відстаней цього графа, якщо ваги ребер і вершин графа дорівнюють 1, має вигляд

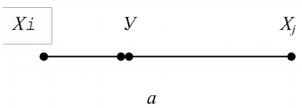
	X_1	X_2	X_3	X_4	X_5	X_6	S_0
X_1	0	1	2	3	2	3	3
X_2	3	0	1	2	1	2	3
X_3	4	3	0	1	2	3	4
X_4	3	2	2	0	1	2	3
X_5	2	1	1	2	0	1	2*
X_6	1	2	3	4	3	0	4
S_t	4	3*	3*	4	3*	3*	

Центр і радіус графа

Вершина X_0^*, для якої $S_0(X_0^*) = \min_{X_i \in X} [S_0(X_i)]$, називається зовнішнім центром графа G, аналогічно вершина X_t^*, для якої $S_t(X_i^*) = \min_{X_i \in X} [S_t(X_i)]$, називається внутрішнім центром графа *G*. У графа може бути кілька зовнішніх і внутрішніх центрів відповідно. Число зовнішнього поділу вершини X_0^*, яка є зовнішнім центром, називається зовнішнім радіусом $\rho_0 = S_0(X_0^*)$; число внутрішнього поділу називається внутрішнім радіусом $\rho_t = S_t(X_i^*)$. У графа, зображеного на рис. 2.21, є тільки один зовнішній центр – вершина X_5 і чотири внутрішніх центри, що утворюють безліч $\{X_2, X_3, X_5, X_6\}$. Зовнішній радіус графа дорівнює 2, а внутрішній 3.

Абсолютний центр графа

Ми визначили числа поділу для будь-якої вершини Xi∈X. Узагальнимо це визначення для точок, які можна розміщати на ребрах графа. Отже, якщо a=(Xi, Xj) представляє ребро графа з вагою Cij, то точка B, що поміщається на ребрі, може бути визначена за допомогою задавання довжини l(Xi, Y) ділянки (Xi, Y), причому повинна виконуватися рівність l(Xi, Y) + l(Y, Xj) = Cij.



a

Числа поділу $S_0(X_i)$ і $S_t(Y)$ точки *Y* незалежно від того, чи є вона вершиною графа *G* або точкою ребра графа *G*, визначається так:

$S_0(Y) = \max_{X_i \in X} [v_i d(Y, X_i)]$.

Точка y_0^*, для якої $S_0(X_i) = \min_{Y \subseteq G} [S_0(Y)]$, називається абсолютним зовнішнім центром графа, і аналогічно визначається y_t^* – абсолютний внутрішній центр. Число зовнішнього поділу абсолютного зовнішнього центра називається абсолютним зовнішнім радіусом $r_0 = S_0(в_0^*)$, а число внутрішнього поділу абсолютного внутрішнього центра називається внутрішнім радіусом $r_t = S_t(y_t^*)$.

Приклад 2.3. Розглянемо неорієнтований граф G (рис. 2.22), де всі ваги вершин і ребер дорівнюють 1. Оскільки граф неорієнтований, то числа поділу однакові для однієї й тієї самої вершини, а матриця відстаней симетрична відстані головної діагоналі:

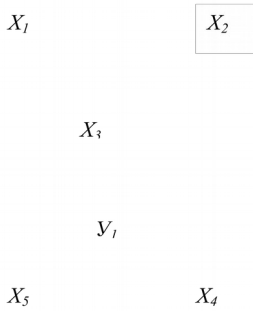


Рис. 2.22 – Граф G

	X_1	X_2	X_3	X_4	X_5
X_1	0	1	2	2	1
X_2	1	0	1	1	2
X_3	2	1	0	1	2
X_4	2	1	2	0	1
X_5	1	2	1	1	0

.

Як бачимо, центром є кожна вершина й радіус дорівнює 2. Якщо тепер вибрати точку Y_1 на ребрі (X_3, X_5) так, що $l(X_5, Y_1) = 1/2$ і, отже, $l(Y_1, X_3) = 1/2$, то відстань від цієї точці до вершин графа задається так:

	X_1	X_2	X_3	X_4	X_5
Y_1	1,5	1,5	1,5	1,5	1,5

.

Отже, число поділу дорівнює 1,5. У такий спосіб точка Y_1 більш центральна, ніж будь-яка вершина графа. Точка Y_1 є абсолютним центром графа так само, як і всі точки, розташовані в середині ребер (X_1, X_2), (X_2, X_3), (X_5, X_4), (X_1, X_2), (X_4, X_2), (X_5, X_1). Жодна з вершин графа не є абсолютним центром. У загальному випадку може бути кілька абсолютних центрів, розташованих у вершинах графа і на ребрах графа.

2.8. Оцінка пропускної здатності мережі

Нехай мережа задана графом $G(V, E)$ з одним джерелом s і одним одержувачем інформації t. Виникає задача визначення величини максимального потоку інформації, який можна передати між вузлами s і t. Кожну лінію зв'язку в мережі будемо характеризувати пропускною здатністю c_{ij}, а потік будемо задавати цілочисельною функцією f_{ij}, заданою на безлічі E, при цьому

$$0 \le f_{ij} \le c_{ij} \tag{2.7}$$

і припускаємо, що при передачі інформації в кожний вузол мережі приходить деякий потік f і такий самий потік з нього виходить. Останнє можна формально записати так:

$$\sum_i f_{ij} - \sum_h f_{jh} = \begin{cases} -f \text{ если } j = s, \\ 0 \text{ если } j \ne s, t, \\ +f \text{ если } j = t. \end{cases} \tag{2.8}$$

Тоді задача визначення величини максимального потоку інформації, який можна передати між заданою парою вузлів мережі, можна сформулювати як задачу лінійного програмування, у якій потрібно максимізувати цільову функцію

$$F = \sum_j f_{sj} \tag{2.9}$$

при виконанні обмежень (2.7), (2.8). Величина максимального потоку в мережі тісно пов'язана з поняттям розрізу мережі, заданої графом $G(V, E)$. Розрізом $(X, \overline{X})$ у графі $G(V,E)$ будемо називати безліч ребер $\{i, j\}$, для яких або $i \in X, j \in \overline{X}$, або $j \in X$, і $\in \overline{X}$, де X – деяка підмножина вузлів мережі, $\overline{\tilde{O}}$ - доповнення підмножини X до V. Тобто розріз являє собою безліч ребер, видалення яких у мережі перетворює мережу в незв'язну (вона може розпастися на кілька незв'язних компонентів). Пропускною здатністю розрізу або його величиною називають суму

$$C(X, \overline{X}) = \sum_{i,j} C_{ij} . \tag{2.10}$$

У загальному випадку сума у виразі (2.10) береться по всіх ребрах з'єднуючої підмножини X і $\overline{X}$. Розріз із мінімальною пропускною здатністю називається мінімальним розрізом. Оцінка пропускної здатності мережі базується на теоремі «про максимальний потік і мінімальний розріз».

Теорема 2.3. У будь-якій мережі величина максимального потоку з джерела s у вузол-одержувач t дорівнює пропускній здатності мінімального розрізу.

Для аналізу пропускної здатності двополюсних мереж більш зручним є поняття *s–t*–перетину мережі. *S–t*–перетином мережі називається ненадлишкова сукупність ребер, видалення якої з мережі обриває всі шляхи з вершини s в t. Так, для мережі, наведеної на рис. 2.23, стосовно вузлів $s = 1$ і $t = 3$ можна виділити чотири перетини, утворені ребрами (*a, d*); (*b, h*); (*a, c, h*); (*d, c, b*). На рис. 2.23 поруч із символами ребер стоять значення їхніх пропускних здатностей.

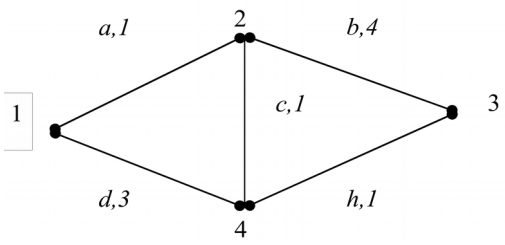


Рис. 2.23. Граф мережі

Кожен *s–t*–перетин має величину, обумовлену сумою пропускних здатностей ребер, що утворюють даний перетин. Так, перераховані перетини мають такі величини пропускних здатностей: $C(a, d)=1+3=4$; $C(\text{b, h}) = 4 + 1 = 5$; $C(a, c, h) = 1 + 1 + 1 = 3$; $C(d, c, b) = 3 + 1 + 4 = 8$.

З теореми про максимальний потік і мінімальний розріз випливає важливий *наслідок:* величина максимального потоку, який можна передати з вузла s в t, дорівнює пропускній здатності мінімального *s–t*–перетину. *S–t*–перетин є окремим випадком розрізу й відрізняється від нього тим, що видалення ребер *s–t*–перетину розбиває мережу на два незв'язні компоненти, а видалення ребер, що утворюють розріз, може розбити мережу на кілька незв'язних компонентів.

Оцінка пропускної здатності двополюсної мережі заснована на побудові двійкової булевої функції, що дозволяє перелічити всі *s–t*–перетини й складається з таких етапів:

1. За структурною матрицею мережі визначається безліч шляхів mst між заданою парою вершин *s* і *t*.

2. Безліч шляхів записується як диз'юнкція добутків символів ребер, утворюючих кожен зі шляхів розглянутої безлічі (при цьому знаки інверсії над символами ребер, що визначають порядок проходження ребра в структурній матриці).

3. Кожний доданок розміщення в дужках і всі знаки диз'юнкції заміняються знаками кон’юнкції (множення).

4. Розкривають всі дужки й відповідно до законів булевої алгебри приводять подібні, використовуючи формулу поглинання $xy \vee x = x$.

Кожний доданок отриманого виразу буде представляти один з можливих *s–t*–перетинів, а всі доданки утворюють перелік всіх поглинених не *s–t*–перетинів.

5. Визначаємо величини кожного з перетинів і серед них вибираємо мінімальний, що і визначає величину максимального потоку, який можна передати з s в t.

Примітка. Перед розкриттям дужок для спрощення виразів доцільно скористатися формулою об'єднання $(a \vee x)(a \vee y)=(a \vee xy)$.

Приклад 2.4. Визначимо величину максимального потоку інформації, який можна передати з вузла 1 мережі, наведеної на рис. 2.23, у вузол 3.

Складемо структурну матрицю мережі:

$$B = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ 1 & 1 & a & 0 & d \\ 2 & \bar{a} & 1 & b & c \\ 3 & 0 & \bar{b} & 1 & h \\ 4 & \bar{d} & \bar{c} & \bar{h} & 1 \end{array}.$$

Викреслюючи перший стовпець і третій рядок у матриці B, одержимо визначник, розкриття якого дозволить перелічити всі шляхи з вузла 1 у вузол 3:

$$m_{13} = \begin{vmatrix} a & 0 & d \\ 1 & b & c \\ \bar{c} & \bar{h} & 1 \end{vmatrix} = a\begin{vmatrix} b & c \\ \bar{h} & 1 \end{vmatrix} \vee\ d\begin{vmatrix} 1 & b \\ \bar{c} & \bar{h} \end{vmatrix} = a(b \vee \bar{h}c) \vee\ d(\bar{h} \vee b\bar{c}) = ab \vee a\bar{h}c \vee d\bar{h} \vee dbc.$$

Ми одержали перелік шляхів, що ведуть з вершини 1 у вершину 3. Далі, прибираючи знаки інверсії, перепишемо перелік шляхів $m_{13} = ab \vee ahc \vee dh \vee dbc$.

Заміняючи всі знаки диз'юнкції на кон'юнкції й навпаки, одержимо $S_{13} = (a \vee b)(a \vee h \vee c)(d \vee h)(d \vee b \vee c)$.

Поєднуючи перші дві дужки по a, а другі дві по *d*, одержимо $S_{13} = (a \vee hb \vee bc)(d \vee bh \vee hc)$.

Дві дужки, що залишилися, можна об'єднати по *hb*, тоді одержимо

$$S_{13} = (hb \vee (a \vee bc)(d \vee hc)) = hb \vee ad \vee ahc \vee bcd \vee bhc =$$

$$= hb \vee ad \vee ahc \vee bcd.$$

Так, ми одержали чотири перетини, величини яких відповідно дорівнюють $C(hb) = 5;\quad C(ad) = 4;\quad C(ahc) = 3;\quad C(bcd) = 8$.

Мінімальним по величині, що дорівнює 3, є перетин ahc, отже величина максимального потоку інформації, який можна передати з вершини 1 в 3, дорівнює 3 умовним одиницям.

Графічний метод Форда–Фалкерсона

Графічний метод застосуємо тільки для планарних мереж, його доцільно використовувати для попередніх розрахунків мереж з невеликою кількістю вузлів.

1. Для визначення пропускної здатності мережі її зображують на площині у вигляді графа так, щоб ніякі ребра графа не перетиналися одне з одним у точках, відмінних від вершин.

2. Потім у графі вибирається самий верхній по контуру графа шлях між заданими вершинами.

3. В обраному шляху знаходять ребро з мінімальною пропускною здатністю. Запам'ятовують її й віднімають із всіх пропускних здатностей ребер графа, що входять у цей шлях.

4. Ребро з пропускною здатністю, що дорівнює 0, виключається з розгляду.

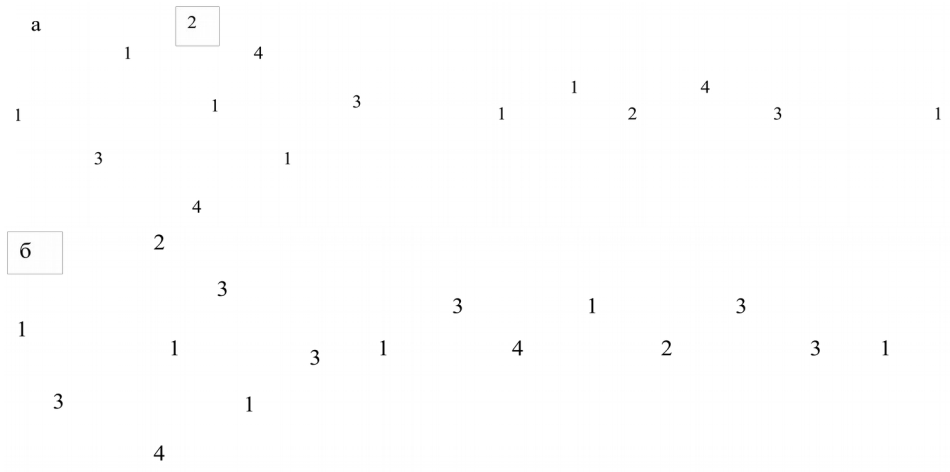
5. Виконання пунктів 2–4 триває доти, поки всі шляхи між заданою парою вершин не будуть обірвані.

6. Суми пропускних здатностей ребер, які запам'ятовуються, при аналізі з кожного шляху визначають максимальну пропускну здатність між заданою парою вершин.

Примітка. При розгляді кожного наступного шляху мінімальне ребро вибирається з урахуванням змін пропускних здатностей ребер, зроблених вирахуванням мінімальних пропускних здатностей у попередніх ітераціях.

Приклад 2.5. Потрібно визначити максимальну пропускну здатність мережі (рис. 2.24, а) між вершинами 1 і 3.

Етапи розв’язання задачі визначення максимальної пропускної здатності мережі графічним методом показано на рис. 2.24, а - в. Таким чином, максимальна пропускна здатність мережі між вузлами 1–3 дорівнює сумі пропускних здатностей, які запам'ятовувалися при аналізі шляхів (вони відзначені окремими кружками праворуч від рисунка) і становить 1 + 1 + 1 = 3 умовні одиниці.



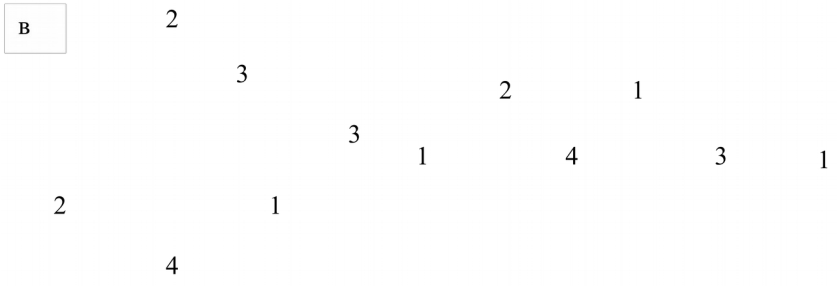


Рис. 2.24. Графи мережі

2.9. Задачі визначення найкоротших шляхів

Нехай заданий граф мережі $G(V,E)$ і задана матриця довжин $L=|l_{ij}|$, тоді задача про НШ полягає в знаходженні найкоротшого шляху з початкової вершини s до заданої кінцевої вершини t за умови $t(R(s))$, де $R(s)$– безліч вершин, яких можна досягнути з s в G. Всі відомі методи розв'язання цієї задачі засновані на твердженні, яке можна сформулювати так:

Твердження 2.5. Якщо найкоротший шлях $\overset{*}{\mu}_{it}$ від довільного вузла і до вузла t проходить через проміжні вузли X_1, X_2, ..., $X_к$, то шляхи $\mu_{X_1t}, \mu_{X_2t}.....\mu_{X_kt}$ є найкоротшими шляхами від вузла X_i до вузла t.

Доведення. Нехай шлях $\mu_{it}(i, X_1, X_2, ..., X_к, t)$ в G з і в t є найкоротшим. Припустимо, що твердження для шляху $\mu_{2t}(X_2,....X_k,t)$ не виконується, тобто існує більш короткий шлях $\overset{*}{\mu}_{2t}$, але тоді це суперечить первісному припущенню про те, що шлях μ_{it} є найкоротшим, отже наше припущення про те, що твердження для шляху μ_{X_it} не виконується, не є правильним. Оскільки таке міркування можна провести для будь–якого вузла мережі $X_i \in \mu_{it}$, то по індукції випливає справедливість даного міркування для довільного відрізка шляху μ_{X_it}.

Існуючі методи визначення найкоротших шляхів можна умовно розділити на дві основні групи:

- методи, що використовують алгоритми нумерації вузлів і галузей мережі (індексні методи);
- методи, що використовують операції над матрицями (матричні методи).

2.9.1. Індексні методи

У матриці довжин $L = |l_{ij}|$, що задає граф мережі, елемент $l_{ij} = \infty$, якщо між вузлами i і j мережі не існує безпосередньої лінії зв'язку, що з'єднує ці вузли.

Алгоритм Форда – Фалкерсона

Крок 1. Всім вершинам, крім фіксованої вершини *s,* припишемо вагу $v_j = \infty$, фіксованій вершині s припишемо вагу $v_s = 0$ (*s* – це вершина, з якої ми хочемо визначити найкоротші шляхи до всіх інших вершин графа мережі).

Крок 2. Знаходимо в *G* таке ребро (*i, j*), що $l_{ij} + v_i < v_j$; якщо таке ребро знайдено, то $v_j := l_{ij} + v_i$, і продовжуємо виконання кроку 2 дотi, поки не стане можливим знайти таке ребро, для якого нерівність, що перевіряється, виконувалася б.

Рекомендації з виконання кроку 2. З метою зменшення обсягу обчислень операцію перерахування ваг v_j доцільно здійснювати для вершин суміжних з *s*, а потім для вершин, суміжних зі знов позначеними.

Після закінчення виконання кроку 2 всі ваги v_j вершин графа *G* відповідають величинам найкоротших шляхів з вершини s до всіх інших вершин графа.

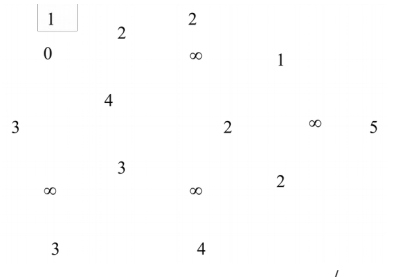
Для ідентифікації шляху, тобто встановлення, через які вершини проходять найкоротші шляхи довжиною v_j, можна використовувати перевірку рівності $v_j = l_{ij} + v_i$ і вершини, що задовольняють рівність, включати в шлях, тобто визначити безліч вершин, що входять у найкоротший шлях від вершини s до вершини j.

Таким чином, задача визначення НШ індексними методами розв’язується у два етапи:

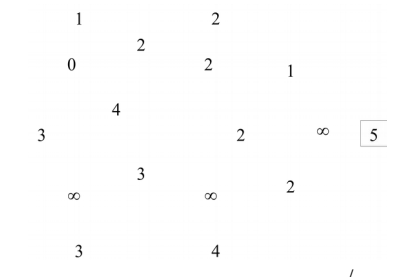
перший етап – це визначення величин НШ від заданої вершини s до всіх інших вершин графа мережі;

другий етап – це ідентифікація НШ від заданої вершини s до всіх інших вершин графа мережі.

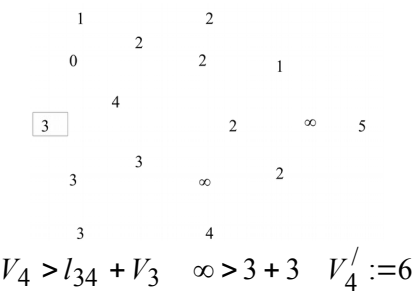
Приклад 2.6



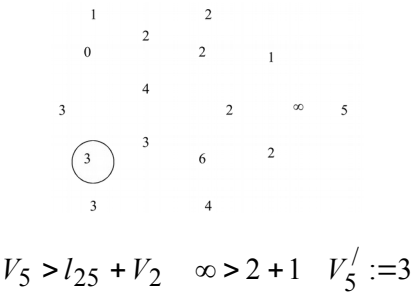
$V_2 > l_{12} + V_1 \quad \infty > 2 + 0 \quad V_2^{/} := 2$



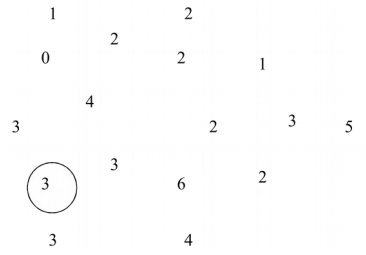
$V_3 > l_{13} + V_1 \quad \infty > 3 + 0 \quad V_3^{/} := 3$

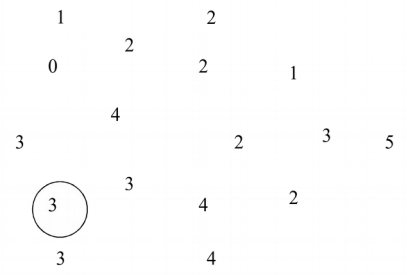


$V_4 > l_{34} + V_3 \quad \infty > 3 + 3 \quad V_4^{/} := 6$



$V_5 > l_{25} + V_2 \quad \infty > 2 + 1 \quad V_5^{/} := 3$





$V_4 > l_{24} + V_2 \quad 6 > 2 + 2 \quad V_4^{/} := 4$

Далі видно, що не існує ребра l_{ij}, для якого виконалася б нерівність $v_j = l_{ij} + v_i$, тому ваги v_j, привласнені вершинам, є величинами НШ.

Визначимо, через які вершини проходить НШ довжиною 3 від вершини 5 до вершини 1. Для цього скористаємося перевіркою рівності $v_j = l_{ij} + v_i$. П'ята вершина суміжна з вершинами 2 і 4, перевіряємо: $V_5 = l_{54} + V_4 \quad 3 \neq 2 + 4$, отже, v_4 не лежить на НШ, аналогічну перевірку робимо для вершини 2: $V_5 = l_{52} + V_5 \quad 3 = 1 + 2$, отже, v_2 лежить на НШ і ми включаємо вершину 2 у шлях. Для вершини 2 починаємо переглядати всі суміжні з нею вершини, які вже ввійшли в шлях (2 суміжна з 4, 3, 1), тому перевіряємо:

$V_2 = l_{42} + V_4 \quad 2 \neq 2 + 4$ – вершина 4 не лежить на НШ;

$V_2 = l_{32} + V_3 \quad 2 \neq 4 + 3$ – вершина 3 не лежить на НШ;

$V_2 = l_{12} + V_1 \quad 2 = 2 + 0$ – вершина 1 лежить на НШ.

Оскільки ми прийшли до вершини, позначеної нулем, то ми закінчили ідентифікацію шляху від вершини 5 до вершини 1.

Загальна складність першого етапу пошуку НШ не перевищує n^3 операцій додавання й порівняння, де n – кількість вершин у графі, загальна складність другого етапу не перевищить n^2 операцій додавання й порівняння. У такий спосіб загальна складність алгоритму не перевищує $O(n^3 + n^2) \approx O(n^3)$ елементарних операцій додавання й порівняння.

Недолік алгоритму: у ньому не вказується стратегія перебору вершин при розміщенні міток, хоча явно існують неефективні стратегії, які збільшують кількість операцій в алгоритмі. Зазначений недолік усувається в алгоритмі Дейкстри для визначення НШ індексним методом.

Алгоритм Дейкстри

У загальному випадку алгоритм заснований на приписуванні вершинам графа тимчасових міток, які дають верхню оцінку довжини шляху від вершини s до i–ї вершини, і на кожному кроці ітерації одна з тимчасових міток стає постійною. Останнє вказує, що мітка вже не є верхньою оцінкою довжини НШ, а дає точну довжину НШ від вершини s до розглянутої вершини t. Опишемо цей алгоритм по кроках.

Присвоєння початкового значення

Крок 1. $V(s) := 0$ і вважати цю мітку постійною. Покласти $V(i) = \infty$ для всіх $i \neq s$ і вважати ці мітки тимчасовими. Покласти $P := s$.

Відновлення міток, яких можна досягнути

Крок 2. Для всіх і, досяжних з *P,* мітки яких тимчасові, змінити відповідно до виразу:

якщо $v_i > l_{pi} + v_p$, то $v_i := l_{pi} + v_p$.

Перетворення мітки в постійну

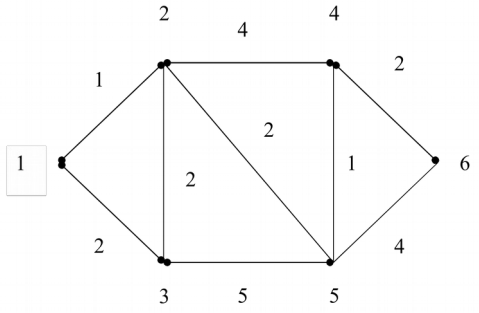
Крок 3. Серед всіх вершин з тимчасовими мітками знайти таку, для якої $\overset{*}{V_i} = \min\limits_i \{V_i\}$.

Крок 4. Уважати мітку вершини $\overset{*}{V_i}$ постійною й покласти $i := p$.

Крок 5. Перевіряємо $p = t$, якщо так, то алгоритм закінчив роботу, якщо ні, то перейти до кроку 2.

Якщо потрібно визначити НШ між s і всіма іншими вершинами, то алгоритм закінчує роботу, коли всі мітки стануть постійними.

Приклад 2.7. Визначити НШ між вершинами 1 і 6.



Виконаємо алгоритм по кроках (зірочкою * будемо позначати мітки, які є постійними).

На початку перша вершина одержує мітку 0 і вона оголошується постійною (після того як мітка стала постійною, вона залишається нею до кінця роботи алгоритму), іншим вершинам привласнюються мітки рівні (і вони є змінними).

1	2	3	4	5	6
0*	∞	∞	∞	∞	∞
0*	1*	2	∞	∞	∞
0*	1*	2*	5	3	∞

З вершини 1 переглядаємо всі суміжні вершини, що мають змінні мітки, ними є вершини 2 і 3, і для них перевіряємо виконання нерівностей
$V_2 > l_{12} + V_1; \quad \infty > 1 + 0; \quad V_2 := 1; \; V_3 > l_{13} + V_1; \quad \infty > 2 + 0; \quad V_3 := 2$.

Вага вершини $V_2 = 1$ є мінімальною, тому мітку вершини 2 оголошуємо постійною і переходимо на вершину 2 в аналізованому графі. Переглядаємо всі вершини, суміжні з вершиною 2, що мають змінні мітки, ними є вершини 4, 5 і 3, і для них перевіряємо виконання нерівностей
$V_4 > l_{24} + V_2; \quad \infty > 4 + 1; \quad V_4 := 5;$
$V_5 > l_{25} + V_2; \quad \infty > 2 + 1; \quad V_5 := 3; \; V_3 > l_{23} + V_2; \quad 2 > 2 + 1; \quad V_3 := 2$.

Мінімальною є вага вершини V_3, тоді V_3=3, оголошуємо мітку постійною і переходимо на вершину 3 досліджуваного графа. Переглядаємо всі вершини, суміжні з вершиною 3, що мають змінні мітки, це - вершина 5, і для неї перевіряємо виконання нерівності
$V_5 > l_{35} + V_3;\quad 3 > 5 + 3;\quad V_5 := 3$.

Оскільки вершина 5 одна, то її мітка V_5= 3 є мінімальною, і ми її оголошуємо постійною і переходимо у вершину 5.

Переглядаємо всі вершини, суміжні з вершиною 5, що мають змінні мітки, ними є вершини 6 і 4 , і для них перевіряємо виконання нерівностей
$V_6 > l_{56} + V_5;\quad \infty > 4 + 3;\quad V_6 := 7;\ V_4 > l_{54} + V_5;\quad 5 > 1 + 3;\quad V_4 := 4$.

Мінімальною є вага вершини V_4 = 4, і ми оголошуємо її мітку постійною і переходимо на вершину 4 досліджуваного графа. Переглядаємо всі вершини, суміжні з вершиною 4, що мають змінні мітки, такою є тільки вершина 6, і для неї перевіряємо виконання нерівності $V_6 > l_{46} + V_4;\quad 7 > 2 + 4;\quad V_6 := 6$.

Вершина 6 одна, її мітка V_6= 6 є мінімальною, і ми її оголошуємо постійною. Оскільки вершина 6 одержала постійну мітку V_6=6, то алгоритм закінчує роботу, оскільки довжина НШ до вершини 6 визначена й дорівнює 6.

2.9.2. Визначення найкоротших шляхів матричними методами

За необхідності визначити найкоротший шлях між всіма парами вершин графа можна використовувати N раз розглянуті в попередньому пункті алгоритми Форда або Дейкстри. При цьому тимчасова складність алгоритмів збільшується в N раз. Виходом з даного положення з'явилися матричні методи, вони дозволяють одночасно відшукувати найкоротші шляхи між всіма парами вершин у графі. Основоположником даних методів вважається Шимбел, який запропонував визначати найкоротші шляхи піднесенням до степеня *r* матриці ваг. При цьому для зведення до степеня матриці *L* пропонується використовувати спеціальні операції:

1. Операція множення двох величин x і y при зведенні матриці в ступінь відповідає їхній алгебраїчній сумі, тобто $x * y = y * x$ відповідає $x + y = y + x$.

2. Сума двох величин дорівнює мінімальному з доданків, тобто $x + y = y + x$ відповідає min (*x, y*).

Зведення матриці в ступінь можна розглядати як послідовне (*r*–1) кратне множення матриці довжин *L* на саму себе

$$L^{\varepsilon} = L * L^{\varepsilon - 1};\ \varepsilon = (\overline{1, r}).$$

Якщо l_{ix} є елементом *I*-рядка й *x*-стовпця матриці *L*, а $l_{xj}^{\varepsilon-1}$-елементом *x*-рядка і *j*-стовпця матриці $L^{\varepsilon-1}$, то за звичайними правилами множення елемент *i*-рядка й *j*-стовпця l_{ij}^{ε} матриці L^{ε} буде дорівнювати

$$l_{ij}^{\varepsilon} = l_{i1}l_{1j}^{\varepsilon-1} + l_{i2}l_{2j}^{\varepsilon-1} + ... + l_{in}l_{nj}^{\varepsilon-1}.$$

З урахуванням операцій, уведених Шимбелом, вираз для l_{ij}^{ε} запишеться так:

$$l_{ij}^{\varepsilon} = \min[(l_{i1} + l_{1j}^{\varepsilon-1}) + (l_{i2} + l_{2j}^{\varepsilon-1}) + ... + (l_{in} + l_{nj}^{\varepsilon-1})],$$

або в скороченому вигляді $l_{ij}^{\varepsilon} = \min\limits_{x=\overline{1,n}}\{l_{ix} + l_{xj}^{\varepsilon-1}\}$.

Фізичний зміст l_{ij}^{ε} при $\varepsilon = n - 1$ – це величина найкоротшого шляху від вершини і до вершини *j*. Слід зазначити, що при зведенні матриці *L* у степінь існує межа, при якій $L^{\varepsilon*} = L^{\varepsilon+1}$(*), тобто подальше множення матриці $L^{\varepsilon*}$ на L не призводить до зміни матриці $L^{\varepsilon*}$. Величина ε^* визначається максимальною кількістю ребер у найкоротшому шляху, а отже, $\varepsilon^* < n-1$, тому як тільки в процесі зведення в степінь матриці починає виконуватися рівність (*), то подальше множення можна припинити, а отримана матриця $L^{\varepsilon*}$ буде матрицею найкоротших шляхів, що називається дисперсійною, і її позначають $L^{\varepsilon*} = D|\delta_{ij}|$.

Метод Оттермана

Подальшим розвитком методу Шимбела є метод, запропонований Оттерманом. Спочатку, користуючись методом Шимбела, визначається дисперсійна матриця, потім зі структурної матриці довжин L утворюється модернізована матриця $M = \|\gamma_{ij}\|$, яка отримується заміною в структурній матриці довжин L значень елементів головної діагоналі з 0 на ∞. Отримана в такий спосіб модернізована матриця помножується на дисперсійну матрицю *D* з використанням операцій, уведених Шимбелом. При множенні матриці $M = \|\gamma_{ij}\|$ на матрицю *D* утворюється матриця

$$\Delta = M*D = \|d_{ij}\|.$$

Елементи даної матриці використовуються для одержання дистанційних матриць (матриць величин 1–го, 2–го й т. д. найкоротших шляхів) і матриць маршрутів, що містять номери вершин обраних найкоротших маршрутів. Кожний елемент d_{ij} матриці Δ з урахуванням уведених операцій буде мати вигляд

$$d_{ij} = \min_{k}[(\gamma_{i1} + \delta_{1j}) + (\gamma_{i2} + \delta_{2j}) + ... + (\gamma_{in} + \delta_{nj})],$$

або в скороченому вигляді

$$d_{ij} = \min_{k}[(\gamma_{ik} + \delta_{kj})],$$

де k – проміжні вузли в шляху.

Кожна сума $(\gamma_{ik} + \delta_{kj})$ визначає довжину шляху від вузла і до вузла j, якщо першим проміжним вузлом після вузла і буде вузол k (рис. 2.25).

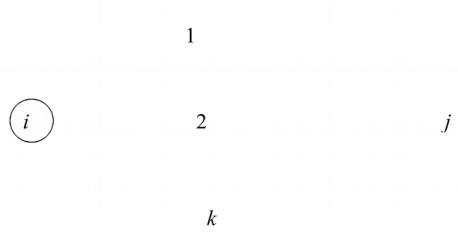


Рис. 2.25. Граф зв'язків між вузлами i і j

Величина найменшого з усіх виразів (сум), що входять в елемент d_{ij}, визначає довжину найкоротшого шляху від вузла і до вузла j

$$d_{ij}^{1} = \min_{1}[(\gamma_{i1} + \delta_{1j})],$$

заноситься як елемент (i, j) у дистанційну матрицю 1–го вибору, а значення 1 цього виразу заноситься як елемент (i, j) у матрицю маршрутів 1–го вибору. Другий за величиною вираз з d_{ij}, що є довжиною другого найкоротшого шляху між вузлами i і j,

$$d_{ij}^{2} = \min_{2}[(\gamma_{i2} + \delta_{2j})]$$

заноситься як елемент (i, j) у дистанційну матрицю 2–го вибору, а 2 у матрицю маршрутів 2–го вибору. Вираз

$$d_{ij}^{K} = \min_{к}[(\gamma_{iк} + \delta_{кj})]$$

заноситься в дистанційну матрицю K–го вибору, а K – у матрицю маршрутів K–го вибору. Елементи матриці Δ, отримані множенням матриці $M = \|\gamma_{ij}\|$ на матрицю $D = |\delta_{ij}|$, містять перелік усіх вузлів мережі між

парами, з якого й здійснюється вибір 1–го, 2–го й т. д. найкоротших шляхів. Використання для матриці $M = \|\gamma_{ij}\|$ замість L викликано необхідністю виключати з розгляду вирази вигляду

$$d_{ij}^{\kappa} = \min_{\kappa}[(\gamma_{ii} + \delta_{ii})],$$

які призводять до запису в матрицю маршрутів *К*–го вибору як елемент (i, j) вузла i. Це буде означати, що рух від вузла i до вузла j має відбуватися через вузол i, що не несе ніякої додаткової інформації.

Наведений вище метод визначення НШ 1–го, 2–го й к–го вибору не виключає можливості появи петель (циклів) в обраних шляхах, тому потрібні спеціальні заходи для їхнього усунення.

Вправи

1. Граф зберігається у вигляді списків суміжних вершин. Скільки операцій потрібно, щоб знайти а) кількість вихідних з даної вершини ребер; б) кількість вхідних у дану вершину ребер?

2. Що відбудеться з матрицею суміжності орієнтованого графа, якщо змінити напрямки стрілок на всіх його ребрах, замінивши кожне ребро (u,v) на ребро (v,u)? Як змінити напрямок стрілок, якщо граф зберігається у формі списків суміжних вершин? Оцініть необхідну для цього кількість операцій.

3. Доведіть, що для монотонно зростаючих функцій $f(n)$ і $g(n)$ функції $f(n) + g(n)$ і $f(g(n))$ будуть також монотонно зростати. Що більше, $\log(\log^* n)$ чи $\log^*(\log n)$, якщо $\log^* n$ – це кількість раз, що треба застосувати функцію log, щоб з n одержати число, що не перевершує 1?

4. Доведіть, що в повному неорієнтованому графі кожне ребро належить рівно n – 2 трикутникам (тобто циклам довжини 3).

5. Доведіть, що граф 2 – хроматичний тільки тоді, коли всі його цикли мають парні довжини.

6. Доведіть, що в будь–якому неорієнтованому графі з одиничними вагами вершин існує такий маршрут між деякою парою вершин, де абсолютний центр графа лежить у середині цього маршруту.

7. Порівняйте алгоритми Дейкстри й Форда–Фалкерсона за тимчасовою складністю. Поясніть, як можна зафіксувати кінець роботи алгоритму Форда – Фалкерсона.

8. Доведіть, що задача визначення мінімального *s–t* перетину в планарному графі може бути зведена до визначення найкоротшого шляху в деякому графі.

РОЗДІЛ 3. Задачі лінійного булевого програмування в телекомунікаційних системах і мережах

3.1. Методи розв'язання задач дискретної оптимізації і їхня класифікація

Задачі дискретної оптимізації і їх постановка. З метою єдиного подання понятійного апарату теорії алгоритмів сформулюємо визначення основних термінів.

Під *масовою задачею* (або просто *задачею)* будемо розуміти деяке загальне запитання, на яке варто дати відповідь. Звичайно задача містить кілька параметрів, або вільних змінних, конкретні значення яких не визначені. Задача *П* визначається такою інформацією:

загальним списком усіх її параметрів;

формулюванням тих властивостей, які має задовольняти відповідь або, інакше кажучи, розв'язання задачі.

Індивідуальна задача I виходить із масової *П*, якщо всім параметрам задачі *П* привласнити конкретні значення. Інакше кажучи, індивідуальна задача оптимізації – це пари (*F, C*), де *F*– довільна множина, область припустимих точок, а *C* – функція вартості, що здійснює відображення $C{:}F \to R$. Потрібно знайти точку $f \in F$, для якої $C(f) < C(y)$ для усіх $y \in F$. Така точка називається *глобальним оптимальним розв'язком*.

Задачі математичного програмування, у яких для усіх (для частини) компонентів припустимих параметрів висуваються вимоги

цілочисельності, називаються задачами *цілочисельного (частково цілочисельного) програмування.*

Задачі цілочисельного програмування, у яких припустимі лише значення компонентів векторних альтернатив 0 або 1, називаються *задачами* цілочисельного *програмування з булевими змінними* або *задачами булевого (бівалентного) програмування.*

Задачі цілочисельного програмування, у яких припустимі лише значення компонентів векторних альтернатив $\overline{(0, k-1)}$, називаються *задачами програмування з k–значними змінними* або *k–значного програмування* (у загальному випадку число k є різним для різних компонентів).

Задачі оптимізації на кінцевих множинах об'єктів, що будуються за тими або іншими правилами комбінаторики, називаються *задачами комбінаторного програмування.*

Всі перераховані задачі оптимізації розв'язуються на відповідних дискретних множинах (у більшості випадків кінцевих). У загальному випадку під *дискретною множиною* розуміється множина, всі точки якої є ізольованими точками даної множини.

Кожна задача комбінаторного програмування в принципі може бути представлена як задача *цілочисельного* (частково *цілочисельного*) програмування. Однак таке подання не завжди є доцільним.

Таким чином, *задача оптимізації* – це множина *I* індивідуальних задач оптимізації. В індивідуальній задачі представлені вихідні дані і є достатньо інформації для одержання розв'язку, у той час як задача оптимізації – це набір індивідуальних задач, породжуваних однаковим способом. Так, якщо ми говоримо про індивідуальну задачу комівояжера, то матриця відстаней задана, якщо ж ми говоримо про задачу комівояжера в цілому, то мається на увазі набір усіх індивідуальних задач, що відповідають всім матрицям відстаней.

Оскільки задачі оптимізації будуть розглядатися в обчислювальному аспекті, то необхідно позначити подання індивідуальної комбінаторної задачі оптимізації на вході обчислювальної машини. Можна, звичайно, виписати всі припустимі розв'язки й для кожного вказати значення *C*. Однак для більшості задач деякі індивідуальні задачі будуть мати непропорційно велику кількість припустимих розв'язків, як це має місце, наприклад, у задачі комівояжера. Надалі будемо припускати, що *F* і *C* задані неявно за допомогою двох алгоритмів A^* і A^{**}. Алгоритм A^* по даному комбінаторному об'єкту *f* і множині параметрів *S* вирішує, чи є *f* множиною розв'язків *F*, обумовленою даними параметрами. У свою чергу A^{**} по даному припустимому розв'язку *f* і іншій множині параметрів *Q* видає значення *C*(*f*). Тоді індивідуальну комбінаторну задачу *I* можна визначити як подання параметрів *S* і *Q* з використанням фіксованого кінцевого алфавіту й деякого стандартного розумного кодування.

Отже, комбінаторна оптимізаційна задача $П$ – це обчислювальна задача, що складається з трьох частин:

1. Множина D_n індивідуальних задач.

2. Для кожної $I \in D_n$ кінцева множина $S_n(I)$ припустимих розв'язків індивідуальної задачі I.

3. Функція m_n, що зіставляє кожній індивідуальній задачі $I \in D_n$ і кожному припустимому розв'язку $\sigma \in S_n(I)$ деяке додатне число - ціле число, називане величиною розв'язку σ.

Інакше кажучи, за даними подання параметрів S і Q для алгоритмів A^* і A^{**} знаходження оптимального припустимого розв'язку називають *оптимізаційним варіантом* даної задачі.

Якщо необхідно за даними S і Q знайти вартість оптимального розв'язку, то такий варіант називають *обчислювальним варіантом* комбінаторної задачі оптимізації. Якщо вартість C просто обчислити, то обчислювальний варіант комбінаторної оптимізації не може бути набагато складніше, ніж оптимізаційний варіант.

Особливо важливий при вивченні обчислювальної складності задачі третій варіант комбінаторної задачі оптимізації. Цей варіант, називаний *варіантом розпізнавання,* має такий вигляд. Для даної індивідуальної задачі I, тобто подання S і Q, і цілого числа L визначити, чи існує такий припустимий розв'язок F, що $C(f) < L$.

На відміну від двох раніше розглянутих варіантів, варіант розпізнавання являє собою запитання, на яке можна відповісти "так" чи "ні". Очевидно, відповісти на це запитання не набагато складніше, ніж розв'язати відповідну обчислювальну задачу, тому що після її розв'язання залишається тільки порівняти оптимальну вартість $C(f)$ з L і видати відповідь "так" у тому випадку, якщо $C(f) < L$.

Таким чином, використовуючи тільки припущення про те, що C легко обчислити, ми встановили, що кожний з варіантів – оптимізаційний, обчислювальний і розпізнавання – не складніше, ніж попередній. Виникає запитання, чи не мають ці варіанти однакову складність. Іншими словами, чи не можна розв'язати обчислювальний варіант, ефективно використовуючи гіпотетичний алгоритм, що розв'язує задачу у варіанті розпізнавання, і чи не можна зробити те саме для оптимізаційного й обчислювального варіантів?

Якщо вартість оптимального розв'язку є цілим числом, алгоритм якого обмежений поліномом від розміру входу, то щораз, коли варіант розпізнавання може бути розв'язаний ефективно, те саме справедливо й для обчислювального варіанта. Слід зазначити, що справедливість припущення про те, що $\log C(f)$ обмежений поліномом від розміру входу, випливає, що для розв'язання обчислювальної задачі можна ефективно використовувати будь–який алгоритм, що розв'язує задачу розпізнавання.

Не відомо загального методу для розв’язання оптимізаційного варіанта задачі з використанням алгоритму для обчислювального варіанта.

Багато задач оптимізації допускають графову постановку й можуть бути ефективно розв’язані за допомогою методів теорії графів. Ці ж задачі добре описуються мовою дискретного програмування, у зв'язку з чим їх можна вважати окремими задачами цього напрямку. Облік специфіки задач, а також формулювання їх мовою теорії графів дозволяє одержувати більш прості методи розв’язання задачі на порядок більше, ніж розміри цих самих задач у загальній постановці дискретного програмування. З іншого боку, інтерпретація на основі графів забезпечує наочність і припустимість постановки досить складних прикладних задач і методів їхнього розв’язання.

Значення цілочисельних і комбінаторних задач оптимізації швидко зростає. Це пояснюється необхідністю пошуку ефективних розв’язків для технічних, технологічних, організаційних і соціальних проблем, які все більше ускладнюються, у багатьох випадках після відповідної формалізації зводяться до задач вибору на кінцевих множинах або до змішаних задач вибору на кінцевих і континуальних множинах.

Перелік прикладних задач цілочисельного й комбінаторного програмування досить різноманітний.

До числа основних типів цих задач, що одержали найбільш широку популярність, належать задачі:

а) про ранець (рюкзак) – визначення оптимального набору неподільних об'єктів, що задовольняють обмеження по об'єму або(i) по вазі, або(і) енергоспоживанню й т. д.;

б) мінімальні покриття в застосуванні до синтезу технічних пристроїв;

в) максимальне інцидентне сполучення в застосуванні до формування організаційних систем;

г) призначення – про оптимальний розподіл завдань між виконавцями й у більш широкій постановці про оптимізацію складу засобів і розподілу задач по ланках організаційно–технічного комплексу;

д) транспортного типу;

е) розміщення – про оптимальне розташування й потужність джерела в мережі;

ж) пошуку оптимальних шляхів у мережах, зокрема пошуку оптимального маршруту обходу n пунктів (задача комівояжера);

и) оптимального складання розкладу при організації технологічних процесів, процесів обслуговування, навчання й ін.

Будь-яка математична модель оптимізації при введенні в обмеження умов цілочисельності стає математичною моделлю цілочисельної оптимізації. Узагальнену модель цілочисельного програмування можна подати в такому вигляді:

$$\begin{array}{l} f(\vec{x}) \to \min\limits_{\vec{x} \in \Delta_\beta} \quad \text{або} \quad f(\vec{x}) \to \max\limits_{\vec{x} \in \Delta_\beta} \\ \Delta_\beta = \{ \vec{x} | g_i(\vec{x}) = 0 (i = 1,2,...,l); \quad g_i(\vec{x}) \le 0 (i = l+1,...,m); \\ \vec{x} = (x_1, x_2, ..., x_n), \; x_j \in \Psi_1^j (j = 1,2,...,h); \quad x_j \in \Psi_2^j (j = h+1,...,n) \}. \end{array} \qquad (3.1)$$

У випадку, якщо у формулі (3.1) $\Psi_1^1 = \Psi_1^2 = ... = \Psi_1^h = \Psi_1 = \Psi_2 = N = \{0,1,...,k-1\}$, маємо модель цілочисельного програмування без обмежень на значення цілих чисел.

Якщо $\Psi_1^j = \{0,1,...,k-1\} (j = \overline{1,h})$, $\Psi_2 = \{0,1,...,k-1\}$, то маємо модель k–значного програмування.

Якщо $\Psi_1^1 = \Psi_1^2 = ... = \Psi_1^h = \Psi_1 = \Psi_2 = \{0,1\}$, то маємо модель програмування з булевими змінними: якщо $\Psi_1^j = (j = \overline{1,h})$ – цілочисельного, а якщо $\Psi_2 = R^1$ – модель частково цілочисельного програмування.

Далі в розд. 4 будуть розглянуті варіанти формалізованого опису відомих практичних задач, які знайшли своє застосування в обчислювальних системах.

3.2. Класифікація методів розв'язання задач дискретної оптимізації

Наявність значних труднощів і специфічних особливостей у розв'язанні задач цілочисельного програмування з БЗ породило велику кількість методів і алгоритмів. Матеріали з даного напрямку є у вигляді окремих статей і містяться в ряді відомих монографій. Однак найбільш повними є огляди в роботі [185]. Із усіх існуючих методів і алгоритмів розв'язання комбінаторних задач виділимо тільки ті, які застосовні до класу задач ЦЛП із БЗ (рис. 3.1).

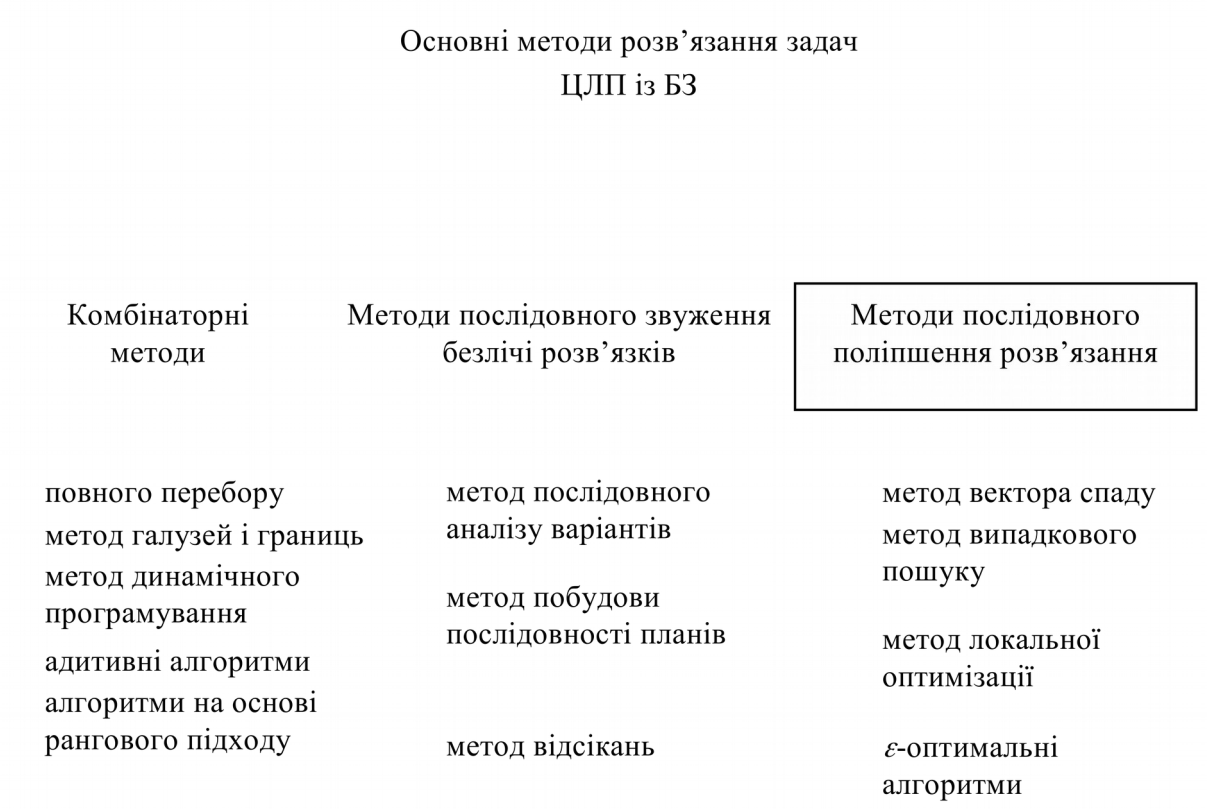


Рис. 3.1. Класифікація методів розв'язання задач ЦЛП із БЗ

Комбінаторні методи. Найбільш простими серед комбінаторних методів є *алгоритми повного перебору*, які виконують перевірку усіх можливих $x = (x_1, ..., x_n) \in 2^n$, одержуючи тим самим точне розв'язання задачі ЦЛП із БЗ. Однак перебір усіх 2^n при $n > 50$ неприйнятний навіть на швидкодіючих ЕОМ. Якщо ж n становить кілька сотень, то кількість потрібних обчислювальних операцій може перевищити поріг Бремермана–Ешбі 10^{100}, що унеможливлює практичне використання даних алгоритмів навіть досить у віддаленому майбутньому.

Адитивні алгоритми, застосовувані для розв'язання задач ЦЛП із БЗ, є найбільш простими й вимагають для своєї реалізації виконання операцій додавання й віднімання (виключаються операції множення й ділення).

Існують різні модифікації адитивних алгоритмів. Однак основна ідея побудови таких алгоритмів полягає в проведенні такого зондування підмножин альтернатив, при якому досягаються результати, еквівалентні повному перебору, але без здійснення останнього й за істотно менший час. З цією метою використовуються ряд прийомів і відповідні їм адитивні чисельні процедури, описані в наступному пункті. Адитивним алгоритмам властиві всі недоліки методів повного перебору, а також якщо в оптимальному розв'язанні 80 – 90 % компонентів вектора $x = (x_1, ..., x_n)$ набувають значення 0 (або 1), то такий розв'язок буде знайдено швидко. Найгіршим випадком для таких алгоритмів є наявність 50 – 60 % нулів (або одиниць), що чергуються між собою. У цьому випадку алгоритм робить майже повний перебір усіх розв'язків.

Найбільш широко використовуваним у цей час є *метод галузей і границь*, уперше сформульований в алгоритмі Ленда й Дойга [76–79]. До переваг такого методу відносять:

порівняну легкість у застосуванні, тому що він не вимагає для своєї реалізації великого об'єму машинної пам'яті;

простоту й видимість обчислювальної схеми;

кінцевість обчислень, що не має потреби в доведенні, оскільки випливає безпосередньо з самої побудови його обчислювальної схеми.

Ефективність розв'язання алгоритмами на основі методу галузей і границь визначається обраними стратегіями розгалуження й підходами до оцінки нижньої (верхньої) границь. Однак, як показують проведені дослідження [75], для точного розв'язання задач ЦЛП виявляються патологічні задачі, об'єм обчислень яких досить близький до повного перебору. У зв'язку з необхідністю розроблення паралельних обчислювальних алгоритмів для розв'язання задач ЦЛП із БЗ була зроблена спроба реалізації алгоритмів, заснованих на методі галузей і границь, на ПОС. При цьому з'ясувалася принципова неможливість такої реалізації, тому що зі збільшенням кількості n процесорних елементів виграш за часом можна одержати лише для $n<n^*$, а при $n>n^*$ час розв'язання починає різко збільшуватися через збільшення об'ємів міжпроцесорного обміну. Тому ставиться задача знаходження оптимальної кількості процесорних елементів n^* при розв'язанні методом галузей і границь.

Метод ДП ґрунтується на принципі оптимальності Р. Беллмана: "оптимальна стратегія має таку властивість: яким би не був початковий стан і початкова стратегія, наступні стратегії (розв'язання) повинні бути оптимальними стосовно поточного стану системи". Сутність динамічного підходу полягає в заміні розв'язання даної n-крокової задачі послідовністю задач: однокрокової, двокрокової й т. д. Оптимальне управління системою на кожному кроці роботи алгоритму не залежить від передісторії процесу, тобто як система перейшла в поточний стан, а визначається тільки самим цим станом. Ці багатокрокові процедури й одержали назву "динамічне програмування" [74].

Важливою перевагою методу ДП є його передбачуваність, тобто можливість теоретично оцінити об'єм обчислювальної роботи й об'єм займаної пам'яті. Однак необхідність запам'ятовування на кожному кроці m-компонентного вектора призводить до потреби у великому об'ємі пам'яті. Ця обставина, образно називана "прокльоном розмірності" [74], становить основні труднощі в застосуванні апарата динамічного програмування.

Останніми в списку комбінаторних методів (рис. 3.1) представлені *алгоритми на основі рангового підходу* до розв’язання задач ЦЛП із БЗ на графах [100–114]. Ранговий підхід до розв’язання задачі 0,1-рюкзак заснований на розбитті *n*-мірного одиничного куба на n рангів, кожний вектор якого має однакову кількість одиниць і нулів. До переваг такого методу, безсумнівно, можна віднести простоту й наочність подання розв’язання задачі; його поліноміальну складність; можливість реалізації на ПОВ.

Однак вимога до рівності коефіцієнтів у функціоналі накладає істотні обмеження на можливість практичного використання запропонованих алгоритмів.

Методи послідовного звуження множини розв’язків. Оптимізаційна схема В.С. Міхайлевича, уведена за назвою *методу ПАВ* [89, 90], причому незалежно й практично одночасно з появою методу галузей і границь, зводиться до повторення таких послідовностей перетворень:

1) розбиття множини варіантів розв’язань задачі на кілька підмножин, кожна з яких має додаткові специфічні властивості;

2) використання специфічних властивостей підмножин розв’язань задач для пошуку логічних протиріч в описі окремих підмножин;

3) виключення з подальшого розгляду тих підмножин варіантів розв’язань, в описі яких є логічні протиріччя.

У цілому метод ПАВ для задачі ЦЛП із БЗ близький до адитивних алгоритмів, тому що він заснований на аналізі системи обмежень задачі. Однак сам аналіз здійснюється різними шляхами.

У загальному випадку метод ПАВ включає (як окремі випадки) схеми методів ДП і галузей і границь. Тому всі недоліки, властиві цим методам, у загальному випадку не усуваються в методі ПАВ.

Помітне застосування в дискретній оптимізації знайшов *метод побудови послідовності планів*, загальний формалізм якого розроблений В.О. Ємеличевим [84, 85]. Ідея цього методу полягає в заміні вихідної екстремальної задачі більше простою, допоміжною задачею й побудові послідовності планів цієї задачі в порядку "погіршення" значень оцінної функції (міноранти або мажоранти).

Серед точних методів розв’язання задач дискретної оптимізації широке поширення одержав *метод відсікання*. Ідея методу така. Спочатку розв’язується задача ЦЛП при відкиданні умови цілочисельності змінних. Одержуваний у такий спосіб розв’язок є оптимальним, якщо він цілочисельний. Якщо ж оптимальний розв’язок нецілочисельний, то до неї

додатково додається нове обмеження, що полягає в такому: множина припустимих розв'язків знову отриманої задачі містить будь-який припустимий розв'язок попередньої задачі й не містить оптимального розв'язку попередньої задачі. Обмеження, що додаються в даному процесі, називають відсіканнями, тому що вони відтинають певну частину множини припустимих розв'язків задачі ЦЛП.

До недоліків методу відсікань належить непередбачуваність його поводження, необхідність обліку все більшої кількості обмежень, слабка пристосованість до розв'язання частково цілочисельних задач, чутливість до помилок округлення, а також слабке застосування для розв'язання задач ЦЛП із БЗ через існування більш ефективних методів точного їхнього розв'язання.

Методи послідовного поліпшення розв'язання. Основною ідеєю методів і алгоритмів послідовного поліпшення розв'язань є така організація пошуку на множині альтернатив, що не звужується, при якій поступово виділялися б більш кращі припустимі розв'язання. Основу цього напрямку становлять наближені методи: метод локальної оптимізації, метод вектора спаду, метод випадкового пошуку.

Метод локальної оптимізації спрямований на побудову ітеративних алгоритмів послідовного поліпшення розв'язань і розроблений у роботах [92–96]. Істотний вплив мали роботи з дискретної математики [78, 83–88, 91, 97–99, 116–119, 121–127, 130–132, 170–185], що призвело до побудови ефективних наближених алгоритмів широкого кола дискретних задач, які з самого початку орієнтуються на існування багатоеКЗтремальних розв'язків. Алгоритми цього методу є вузькоспеціалізованими, і тому не знайшли широкого універсального застосування.

До наближених методів належить *метод вектора спаду*, запропонований І.В. Сергієнком. Застосування цього методу до задач ЦЛП із БЗ дало істотний виграш порівняно з адитивними алгоритмами й алгоритмами, заснованими на методі ПАВ. Недоліком цього методу є істотна залежність точності й збіжності від обраного радіуса наближень.

У розв'язанні задач ЦЛП із БЗ подальший розвиток одержали *методи випадкового пошуку*. Алгоритми цих методів характеризуються тим, що поступове поліпшення досягається шляхом відбору прийнятних альтернатив з послідовності альтернатив, виділюваних з використанням деякого імовірнісного механізму вибору на множині альтернатив. При цьому розрізняють алгоритми керованого й некерованого випадкового пошуку. Застосування цих алгоритмів найбільш доцільно використовувати при побудові початкового наближення для роботи інших алгоритмів. Для задачі ЦЛП із БЗ ефективний алгоритм випадкового пошуку побудований у роботі [130–132], що дає можливість у процесі її розв'язання змінювати функції розподілу ймовірностей і в підсумку зменшувати час одержання прийнятного рішення. Недоліком такого алгоритму є ускладнення

визначення часу виконання в "гіршому" випадку, що не дозволяє його використовувати в масштабі реального часу.

У роботах [84–86, 88] показано, що для задач ЦЛП із БЗ існують поліноміальні асимптотичні оптимальні алгоритми випадкового пошуку, які дозволяють за кінцеву кількість кроків знаходити наближення розв’язків для одномірної задачі. Недоліком методу випадкового пошуку, як і для методу галузей і границь, є принципові труднощі реалізації цих алгоритмів на n-процесорних ПОС. Найбільш повна класифікація сучасних методів розв’язання задач дискретної оптимізації наведена в роботі [187].

Таким чином, з погляду розпаралелювання найкращими є методи, засновані на ранговому підході, яким надалі й буде приділена увага.

Показники ефективності послідовних і паралельних алгоритмів. Основні поняття й визначення, пов'язані з поданням алгоритмів у вигляді графів, дано в роботах [79–81]. Для коректного порівняння розроблюваних алгоритмів з відомими необхідно визначити основні показники їхньої ефективності, що дозволяють проводити порівняльний аналіз.

Найбільш широко застосовуваним показником ефективності роботи алгоритму є відрізок часу, затрачуваний алгоритмом для розв’язання поставленої задачі. Однак такий підхід неоднозначно визначає ефективність алгоритмів, тому що цей показник залежить від типу ЕОМ, на якому виконується алгоритм, від вибору мов програмування й т. д. При аналізі алгоритмів час їхньої роботи буде виражатися в термінах кількості елементарних кроків (арифметичних операцій, порівнянь і т.п.), необхідних для виконання цього алгоритму на гіпотетичній обчислювальній машині. Інакше кажучи, виконання будь-якої операції вимагає одну одиницю часу.

Однак кількість кроків, виконуваних алгоритмом, не однаково для різного розміру входу вихідних даних. Для усунення такої неоднозначності в поводженні алгоритму при переході від одного входу до іншого будемо розглядати всі входи даного розміру n разом і визначимо складність алгоритму для входу як кількість кроків алгоритму в найгіршому разі по усіх входах. У комбінаторних задачах оптимізації входом є комбінаційний об'єкт: граф, множина цілих чисел, сімейство кінцевих множин і т. д. Щоб увести цей вхід в обчислювальній системі для розв’язання, необхідно яким-небудь чином закодувати або подати у вигляді послідовності символів над деяким фіксованим алфавітом, таким як двійковий алфавіт. Кодування комбінаторних об'єктів можна зробити будь-яким з багатьох відомих способів [78–80]. Оскільки вхід алгоритму подається у вигляді послідовності або ланцюжка символів, визначимо розмір входу як довжину цієї послідовності, тобто кількість символів у ній. При аналізі алгоритмів звичайно цікавить швидкість зростання складності алгоритму [79].

Визначення. Нехай $f(n)$ і $g(n)$ – функції, певні на множині цілих додатних чисел, що набувають додатного дійсного значення:

а) позначимо

$$f(n)=O(g(n)), \qquad (3.2)$$

якщо існує така константа $c>0$, що $f(n)\leq Cg(n)$ для досить великих n;

б) позначимо

$$f(n)= O(g(n)), \tag{3.3}$$

якщо існує така константа $c>0$, що $f(n) \geq Cg(n)$ для досить великих n;

в) позначимо

$$f(n)= \Theta(g(n)), \tag{3.4}$$

якщо існують такі константи c, $c'>0$, що $Cg(n)\leq f(n)\leq C'g(n)$ для досить більших n.

Замість виразу (3.4) можна записати

$$f(n)\cup_{\frown} g(n). \tag{3.5}$$

Відношення (3.5) є відношенням еквівалентності. Клас еквівалентності цього відношення, що містить $f(n)$ (тобто множина усіх функцій $g(n)$, таких, що $f(n) = \Theta(g(n))$), називають швидкістю зростання $f(n)$ [323]. Завдяки уведеному поняттю швидкість зростання складності алгоритму, як показано в роботі [323], можна оцінити зверху, використовуючи вираз типу «вимагає часу $O(n^3)$».

Таким чином, часова складність алгоритму відображає витрати часу, що потрібні для цього. Функція, що вхідній довжині n ставить у відповідність максимальний час, не буде повністю визначена доти, поки не зафіксована схема кодування, не обрано обчислювальний пристрій, що визначає час роботи. Той самий алгоритм А можна виконувати за різний час на різних структурах. Тому при переході від однієї структури до іншої часто користуються [325] поняттям ємнісної складності обчислювальної системи, що характеризує зростання апаратурних витрат при такому переході. Апаратурні витрати можуть задаватися кількістю елементарних процесорних елементів, кількістю зв'язків у системі, об'ємом пам'яті й т. д., при цьому використовується вираз «потрібно $O(n^3)$ комірок пам'яті» [79]. При аналізі складності алгоритмів оцінка в найгіршому випадку не завжди об'єктивно несе інформацію про час, затрачуваний алгоритмом на розв’язання задачі, оскільки гірші випадки при розв’язанні конкретної задачі можуть бути рідкими. Тому в роботі для кожного алгоритму теоретично визначена складність у найгіршому випадку, а при експериментальному моделюванні роботи алгоритмів проводяться оцінки їхньої складності в середньому, що дозволяє більш об'єктивно оцінити можливості алгоритму.

Як основні характеристики наближених алгоритмів використовують абсолютну Δf і відносну δf похибки отриманого наближеного розв'язку, які визначаються зі співвідношень

$$\Delta f = \left| f(x) - f(x^*) \right|; \ \delta f = \frac{\left| f(x) - f(x^*) \right|}{f(x^*)}, \tag{3.6}$$

де f – цільова функція, певна на деякій множині M;

x – припустимий розв'язок задачі, що є наближеним;

x^* – оптимальний розв'язок задачі.

Таким чином, як основні критерії ефективності для точних алгоритмів розв'язання задачі ЦЛП із БЗ будемо використовувати функцію часових і апаратурних витрат, для наближених алгоритмів – відносну δf похибку. Показники якості, пов'язані з ефективністю розпаралелювання розроблюваних алгоритмів, будуть уведені безпосередньо при їхньому порівнянні з існуючими.

3.3. Ранговий підхід до розв'язання задачі 0,1-рюкзак

3.3.1. Формальна модель n-мірного одиничного куба в ранговому підході

Розглянемо сутність рангового підходу до задачі ЦЛП з БЗ на прикладі задачі про рюкзак. Загальна постановка цієї задачі формулюється так. Необхідно знайти вектор $\vec{x}$, що забезпечує масимум функції

$$f(\vec{x}) = \sum_{j=1}^{n} c_j \cdot x_j \tag{3.7}$$

за виконання умов

$$\sum_{j=1}^{n} a_{ij} \cdot x_j \le b_i, \tag{3.8}$$

$$x_j \in \{0,1\}, \qquad i = \overline{1,m}; \qquad j = \overline{1,n}. \tag{3.9}$$

Для спрощення викладу математичної моделі розглянемо одномірну задачу, тобто максимізуємо функціонал

$$f(x) = c_1 \cdot x_1 + c_2 \cdot x_2 + \ldots + c_n \cdot x_n \tag{3.10}$$

при обмеженнях

$$\sum_{j=1}^{n} a_{1j} \cdot x_j \le b, \tag{3.11}$$

$$c_1 \ge c_2 \ge \ldots \ge c_n; \ a_{ij} > 0; c_j > 0; j = \overline{1,n}. \tag{3.12}$$

Відповідно до виразів (3.10) – (3.12) граф G (рис. 3.2) являє собою бінарне дерево усіх розв'язків, кількість яких дорівнює 2^n [1, 4].

У множині $X=\{x_j\}$ усіх векторів розмірності n усі компоненти $x_j \in \{0,1\}$ становлять *множину можливих значень.*

Деяка його підмножина V, усі вектори якої задовольняють вирази (3.11)–(3.12), утворюють *множину припустимих розв'язків.*

Множина $H \subset V$ є *множиною оптимальних розв'язків* вихідної задачі, якщо для будь-яких векторів $x \in H$ функціонал (3.10) досягає свого екстремального значення.

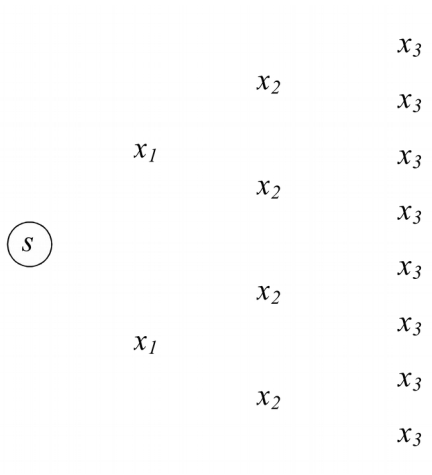


Рис. 3.2. Граф G

Всю множину можливих розв'язків можна розбити на групи векторів, що містять один компонент $x_j=1$, $j=\overline{(1,n)}$, а всі інші дорівнюють 0; два компоненти $x_j=1$ і всі можливі їхні сполучення по 2, а інші дорівнюють 0; три компоненти $x_j=1$ і всі можливі їхні сполучення й т. д. n-компонент $x_j=1$. Якщо позначити підмножини векторів цих груп через m^r $r=\overline{(1,n)}$, тоді множину усіх можливих розв'язків можна записати як об'єднання підмножин m^r

$$X = \bigcup_{j=1}^{n} m^r. \qquad (3.13)$$

Як показано в роботі [4], за графом G можна побудувати граф G' (рис. 3.3), у якому множина шляхів рангу r (*ранг шляхи – кількість ребер, що утворюють шлях*) відповідає групам підмножин, описуваних співвідношенням (3.13). Для цього вершину s з'єднаємо спрямованими ребрами з вершинами 1,2,...,n і т. д.; вершину i з'єднуємо з вершинами i+1,...,n. В останню вершину n входять ребра, спрямовані з усіх вершин, і жодне ребро з цієї вершини не виходить.

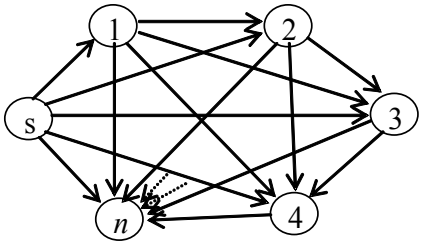


Рис. 3.3. Граф G'

Дерево шляхів $D\Delta$ графа G' з вершини s будується так [4]: на нульовому ярусі (r=0) розташуємо вершину s. На першому ярусі розмістимо всі вершини графа G', що мають зв'язок з вершиною s, і з'єднаємо їх з s (при цьому утворилася підмножина шляхів рангу r=1). У другому ярусі розміститься всі вершини, що мають зв'язок з вершинами першого ярусу, без вершини з номером 1 і з'єднаємо їх з вершинами першого ярусу (утворені всі шляхи рангу r=2) і т. д. доти, поки в останньому не залишиться одна вершина n. На рис. 3.4 n = 4.

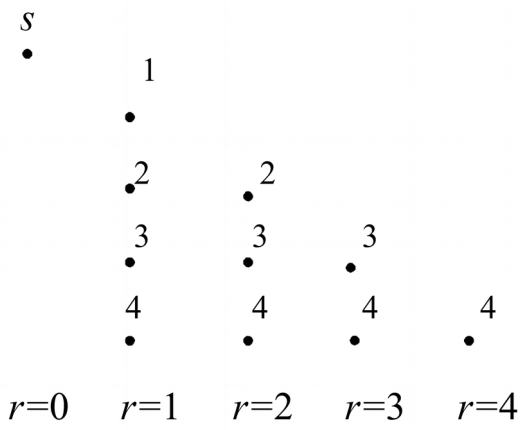


Рис. 3.4. Граф DΔ

Геометрично вершина k графа $D\Delta$ рангу r – це множина векторів x (x_1,x_2,...,x_k,...,x_n), у яких x_k = 1, а на позиціях від 1 до k перебуває r одиниць (рис. 3.5). Ребру, що входить у вершину k графа DΔ, відповідає одиничний вектор e_k (0, 0, …,0,1,0,…0), n–мірному одиничному кубі B^n з одиницею в k–й позиції μ_{sj}^{r} рангу r у графі $D\Delta$ відповідає вектор x, що дорівнює сумі одиничних векторів ребер, по яких він досягнув вершини j рангу r, починаючи з вершини s. Наприклад, шляху $\mu_{s24}^{r=2}$ відповідає вектор $\overrightarrow{x_{s24}}$, що утворюється сумою нульового вектора 0 {0000} і одиничних векторів e_2 = {0100}, e_4 = {0001}, тобто $\overrightarrow{x_{s24}}$ = 0 {0000} + e_2 {0100} + e_4 {0001} = {0101).

Нехай у графі $D\Delta$ кожному ребру, що входить у вершину j, $j=\overline{(1,n)}$, відповідає дві ваги: вага c_j, що дорівнює коефіцієнту при x_j у функціоналі (3.10), і вага a_{1j}, що дорівнює коефіцієнту при x_j в обмеженні (3.11). Тоді шлях μ_{sj}^r у графі $D\Delta$ з вершини s у вершину j характеризується двома довжинами: $d_c\left(\mu_{sj}^r\right)$ – довжиною по вагах функціонала й $d_a\left(\mu_{sj}^r\right)_1$ – довжиною по вагах обмежень.

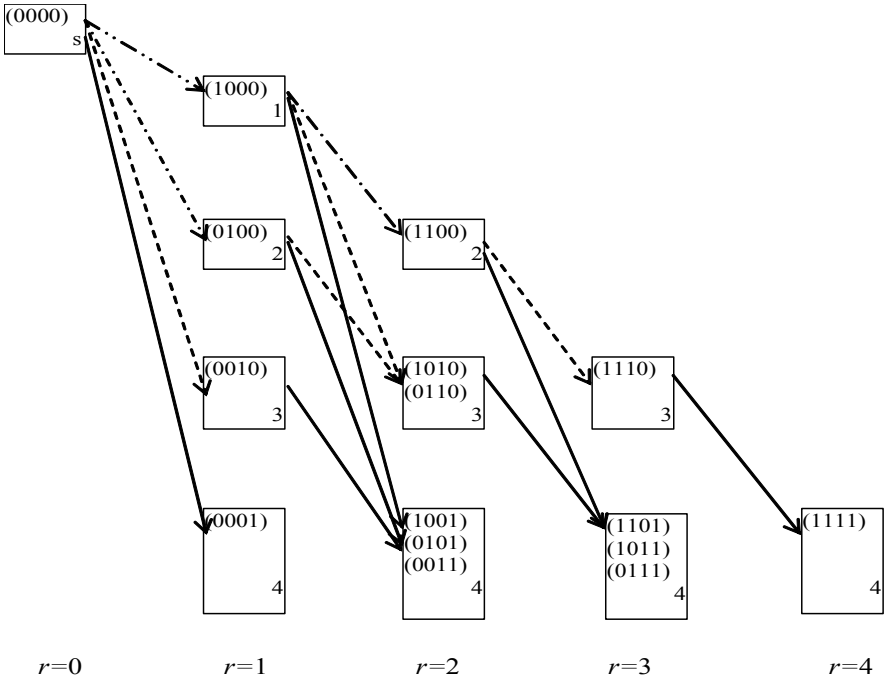


Рис. 3.5. Геометрична інтерпретація графа DΔ

Множина шляхів $m_s^r(j)$ у графі $D\Delta$ до вершин j, розташованих на ярусах $r=\overline{(1,n)}$ від вершини s, можна подати як

$$m_s^r(j)=m_{sj}^{r=1}\cup m_{sj}^{r=2}\cup\ldots\cup m_{sj}^{r=n},\qquad j=\overline{(1,n)},\tag{3.14}$$

де m_{sj}^r – множина шляхів у графі $D\Delta$ від вершини s до вершин j, розташованих на r–х ярусах графа $D\Delta$ (ранг шляху $\mu_{sj}^r\in m_{sj}^r$ визначається кількістю ребер, що утворюють цей шлях).

Варто мати на увазі, що множина шляхів $m_{sj}^{r=k}$ у графі $D\Delta$ відповідає множині векторів $\{\underset{\sqcup}{x}_1,\underset{\sqcup}{x}_2,\ldots,\underset{\sqcup}{x}_v\}$, що містять k одиниць. Отже, $\left|m_{sj}^r\right|=C_n^{r=k}$, тобто кожному шляху в множині $m_{sj}^{r=k}$ відповідає деякий вектор $(x_1,x_2,\ldots,x_n)$. З виразу (3.14) випливає

$$\left|m_s^r(j)\right| = C_n^{r=1} + C_n^{r=2} + \ldots + C_n^{r=n} = 2^n - 1. \qquad (3.15)$$

Таким чином, граф $D\Delta$ являє собою впорядкований по рангах еквівалент n-мірного одиничного куба B^n, у якому шляхи $\mu_{sj}^r \in m_{sj}^r$ відповідають вершинам B^n. Довжина кожного шляху за вагою функціонала визначає значення функціонала (3.10) у вершинах одиничного куба B^n. Довжина по вагах обмежень визначає, чи відповідає дана вершина B^n обмеженням (3.11), тобто чи належить вершина n-мірного одиничного куба B^n гіперплощині (3.11). Якщо $d_a\left(\mu_{sj}^r\right)_1 \le b,$ то вершина належить гіперплощині (3.11) і будемо говорити, що шлях μ_{sj}^r задовольняє властивість ν. Якщо $d_a\left(\mu_{sj}^r\right)_1 > b,$ то вершина n-мірного куба, що відповідає шляху μ_{sj}^r, не належить гіперплощині (3.11), а шлях μ_{sj}^r уважаємо таким, що не задовольняє властивість ν.

Оптимальному розв'язку задачі (3.10)–(3.12) в D відповідає найдовший шлях по вагах функціонала, що задовольняє властивість.

У випадку m-мірної задачі (3.7)–(3.9) ребрам, що входять у вершини графа $D\Delta$, крім ваги c_j функціонала, відповідає m ваг a_{ij} обмежень, а шлях μ_{sj}^r характеризується довжинами: $d_c\left(\mu_{sj}^r\right)$ – довжиною по вагах функціонала й $d_a\left(\mu_{sj}^r\right)_i,\ i = \overline{(1,m)}$ – довжинами по вагах m обмежень.

3.3.2. Принцип оптимізації по напрямку в n-мірному одиничному кубі й узагальненої процедури розв'язання ЦЛП із БЗ на основі ідеї рангового підходу

На основі розглянутої математичної моделі рангового підходу для побудови алгоритмів розв'язання задач ЦЛП із БЗ покладено принцип оптимізації по напрямку в дискретному просторі станів, заданому графом $D\Delta$ [106]. Подання n–мірного одиничного куба у вигляді графа $D\Delta$ дозволяє розбити множина усіх шляхів графа $D\Delta$ з нульової вершини s на Ω локальних областей, де $|\Omega|$ не перевищує величину $\frac{n^2}{2}$, оскільки кількість вершин у графі $D\Delta$ визначається сумою чисел натурального ряду

$$\Omega = n + (n-1) + \ldots + 1 = \frac{n \cdot (n+1)}{2} \approx \frac{n^2}{2}, \qquad (3.16)$$

причому Ω–області в графі $D\Delta$ упорядковані по рангах і шляхи наступного рангу можуть бути отримані на основі шляхів попереднього рангу за

рахунок приєднання до них ребра (j, p) у графі $D\Delta$

$$m_{sp}^{r=r+1} = \left| \left(\forall \left(\mu_{sj}^{r} \in m_{sj}^{r} \right) \right) \cup (j, p) \right|.$$

Нехай задано деякі правила відсікань $\{L_w\}$ шляхів μ_{sj}^{r} у множинах m_{sj}^{r}. Тоді якщо в множинах містяться шляхи, що задовольняють властивість ν і правила $\{L_w\}$, то під *оптимізацією по напрямку* в графі DΔ до вершини p будемо розуміти формування множин $m_{sp}^{r=r+1}$ наступного рангу, які отримують за рахунок виділення в m_{sj}^{r} шляхів, приєднання до яких ребра (j, p) дозволить у множині $m_{sp}^{r=r+1}$ одержати шляхи, що задовольняють правила $\{L_w\}$ на основі такого рекурентного співвідношення:

$$\forall \left(\mu_{sj}^{r} \in m_{sj}^{r} \right) \left| \mu_{sp}^{r=r+1} = L_w \left| \mu_{sj}^{r} \cup (j, p) \right| \right|; \mathrm{p} = (\overline{\mathrm{r}+1, \mathrm{n}}); \mathrm{j} = (\overline{\mathrm{r}, \mathrm{n}}), \quad (3.17)$$

де $\mu_{sj}^{r} \cup (j, p)$ – шлях з вершини s графа $D\Delta$ у вершину p, що проходить через проміжну вершину j і задовольняє правила $\{L_w\}$, тобто одержуваний за рахунок приєднання до шляху μ_{sj}^{r} ребра (j, p), якщо таке з'єднання не суперечить правилам $\{L_w\}$. Надалі для спрощення викладу, якщо шлях $\mu_{sp}^{r=r+1} = \mu_{sj}^{r} \cup (j, p)$ задовольняє правила $\{L_w\}$, будемо говорити, що він задовольняє й властивість ν.

Таким чином, для розв'язання задачі (3.7)–(3.8), використовуючи правила $\{L_w\}$ і оптимізацію по напрямку (3.17), побудуємо деяку узагальнену процедуру A_0, що дозволяє формувати множини локальних екстремумів Ω і виділяти серед них глобальний.

Уведемо узагальнену процедуру A_0, що дозволяє на основі обраного правила відсікань $\{L_w\}$ розв'язувати задачу (3.7) – (3.9).

Узагальнена процедура A_0

Крок 1. З вершини s будуються множини шляхів $m_{sj}^{r=1}$, $j = \overline{(1, n)}$, що задовольняють властивість ν. Виділяються шляхи $\overset{*}{\mu}_{sj}^{r=1}$, що визначають локальні екстремуми областей Ω_j.

Крок 2. Формуються множини шляхів $m_{sp}^{r=r+1}$ $p = \overline{(r+1, n)}$ наступного рангу, що задовольняють властивість ν, на базі множин шляхів m_{sj}^{r} попереднього рангу відповідно до рекурентних співвідношень (3.17). В утворених множинах $m_{sp}^{r=r+1}$ здійснюється відсікання шляхів відповідно до

обраного правила відсікань $\{L_w\}$ і виділяються шляхи $\mu_{sp}^{*r=r+1}$, що визначають локальні екстремуми областей Ω_p.

Крок 3. Перевіряємо $m_{sp}^{r=r+1}$, чи всі множини наступного рангу порожні. Якщо це так, то переходимо до кроку 4, якщо ні, то перевіряємо $r=(n-1)$. У випадку виконання рівності переходимо до кроку 4, інакше збільшуємо r на 1 і виконуємо крок 2.

Крок 4. Виділяємо серед множин локальних екстремумів $\Omega_{j\ j=\overline{(1,n^2/2)}}$ глобальний і процедура A_0 закінчує роботу.

Узагальнена процедура A_0 дозволяє визначити локальні екстремуми в Ω–областях графа $D\Delta$ щораз на кроці 2 і потім на кроці 4 виділяти глобальний екстремум з $n^{2/2}$ локальних, отриманих на основі принципу оптимізації по напрямку (3.17) з використанням правил відсікань, що вводяться, $\{L_w\}$ шляхів у m_{sj}^r множинах.

Таким чином, із представленої математичної моделі n-мірного одиничного куба B^n у вигляді графа $D\Delta$ і сформульованого принципу оптимізації по напрямку на основі рангового підходу випливають такі задачі:

1. Визначення стратегій відсікань $\{L_w\}$ безперспективних шляхів у множинах m_{sj}^r, що призводять до наближених і точних розв'язань задачі ЦЛП із БЗ (3.7) – (3.9).

2. Побудова наближених і точних алгоритмів на основі обраних правил відсікань $\{L_w\}$ для розв'язання одномірних і багатомірних задач ЦЛП із БЗ.

3. Створення паралельних обчислювальних структур як спеціалізованих пристроїв для розв'язання даного класу задач дискретної оптимізації.

3.3.3. Правила відсікання безперспективних варіантів розв'язків на основі узагальненої процедури рангового підходу

В основу методу відсікання безперспективних варіантів для задачі 0,1-рюкзак покладемо множину стратегій $\{L_w\}$, застосування яких до узагальненої процедури A_0 призведе до побудови алгоритмів розв'язання цієї задачі ЦЛП з БЗ.

Найбільш простою стратегією відсікань L_1 при формуванні $r=r+1$ шляхів наступного рангу в множинах $m_{sp}^{r=r+1}$, на основі процедури A_0, є виділення в m_{sj}^r шляхів масимальної довжини по вагах функціонала c_j, довжини яких по вагах обмежень не перевищують величини b_i, тобто $d_a\left(\mu_{sj}^r\right)_i \le b_i$. У цьому випадку шляхами, що задовольняють властивість ν,

будемо називати шляхи μ_{sj}^{r}, довжини яких по вагах обмежень не перевищують величину b_i. Тоді рекурентне співвідношення (3.14), що відповідає стратегії L_1, набуде вигляду

$$\mu_{sp}^{r:=r+1} = \max_{c_j} \left| \mu_{sj}^{r} \cup (j,p) \right|; \mathrm{p} = (\overline{\mathrm{r}+1,\mathrm{n}}); \quad \mathrm{j} = (\overline{\mathrm{r},\mathrm{n}}). \tag{3.18}$$

Приклад 3.1. Потрібно максимізувати функціонал

$$f(x) = 20x_1 + 18x_2 + 18x_3 + 15x_4 + 10x_5 + 4x_6 + 1x_7$$

при обмеженні $7x_1 + 2x_2 + 5x_3 + 2x_4 + 3x_5 + 1x_6 + 4x_7 \le 12$ у відповідності зі стратегією L_1.

Розв'язання. Побудуємо граф $D\Delta$ (вершину s, що відповідає тривіальному розв'язку задачі, показувати не будемо). Шлях μ_{sj}^{r}, $j = (\overline{r,n})$ задається ідентифікатором, що визначає, через які вершини попередніх рангів він проходить (рис. 3.6).

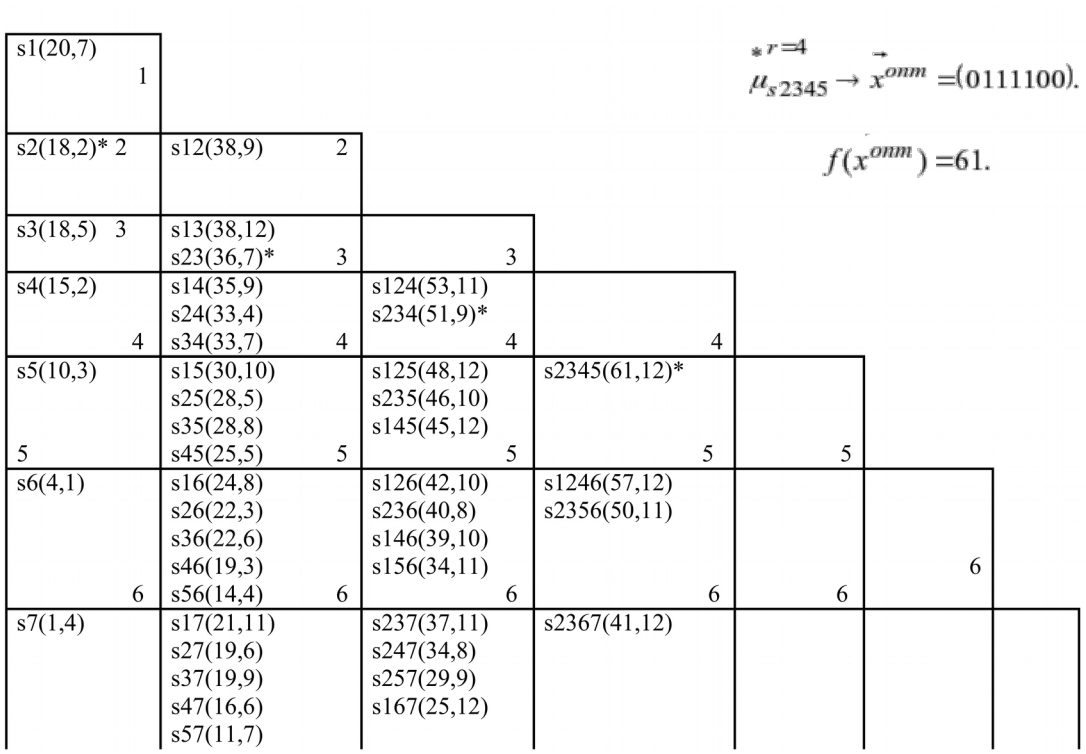


Рис. 3.6. Ілюстрація роботи стратегії L_1

Наприклад, ідентифікатор s146 указує на шлях $\mu_{s146}^{r=3}$, що проходить через вершини $s \to 1 \to 4 \to 6$ у графі $D\Delta$ (рис. 3.4). У дужках на першому місці стоїть довжина шляху по вагах функціонала, а далі m ваг по першому, другому і т. д. m–му обмеженням.

Так, запис "s146(39,10)" відповідає шляху $\mu_{s146}^{r=3}$, що має $d_c\left(\mu_{s146}^{r=3}\right)=39$, а $d_a\left(\mu_{sj}^{r}\right)_1=10$. Шлях μ_{sj}^{r}, що належить оптимальному розв'язку, відзначений "*".

Ітерація 1. Будуємо шляхи рангу r=1, що задовольняють обмеження задачі. Таких шляхів у графі $D\Delta$ сім: $\mu_{s1}^{r=1} - \mu_{s7}^{r=1}$. Серед них вибираємо шлях з найбільшою довжиною по вагах функціонала: $\mu_{s1}^{r=1}$. Позначимо його $\mu_{s1}^{*r=1}$.

Ітерація 2. Будуємо шляхи рангу r=2 відповідно до принципу оптимізації по напрямку (3.17) і стратегії L_1 (3.18). Наприклад, сформуємо шлях $\mu_{s13}^{r=2}$ на основі шляху $\mu_{s1}^{r=1}$, додаючи ребро (1,3). При цьому довжина шляху буде: по вагах функціонала $d_c\left(\mu_{s13}^{r=2}\right)=38$, а по вагах обмеження $d_a\left(\mu_{s13}^{r=2}\right)_1=12$, тобто шлях $\mu_{s13}^{r=2}$ задовольняє обмеження задачі й властивість ν. Аналогічно будуються інші шляхи рангу r = 2. Потім серед усіх шляхів рангу 2 вибираємо шлях з найбільшою довжиною $d_c\left(\mu_{sj}^{r=2}\right)$. Це шляхи: $\mu_{s12}^{*r=2}$ і $\mu_{s13}^{*r=2}$.

Ітерація 3. Будуємо шляхи рангу r=3 на основі шляхів рангу r=2 і виразів (3.17), (3.18). При додаванні до шляху $\mu_{s12}^{r=2}$ ребра (2,3) утвориться шлях $\mu_{s123}^{r=3}$, довжина якого по вагах обмежень $d_a\left(\mu_{s123}^{r=3}\right)_1=14$, що не задовольняє умову задачі. Тому такий шлях не показаний. Якщо в множині $m_{sj}^{r=2}$ кілька шляхів, більший з яких при приєднанні якого-небудь ребра не задовольняє властивість ν, то згідно з виразом (3.17) вибираємо наступний шлях, що належить цій множині, але менший по вагах функціонала. На r=2 (рис. 3.6) це можна спостерігати при побудові шляху з множини $m_{s3}^{r=2}$ в $m_{s3}^{r=3}$ Серед усіх шляхів рангу 3 вибираємо шлях з найбільшою довжиною $d_c\left(\mu_{sj}^{r=3}\right)$. Це шлях $\mu_{s124}^{*r=3}$.

Ітерація 4. Будуємо шляхи рангу r=4, аналогічно описаним в ітерації 3. Серед усіх шляхів рангу 4 також вибираємо шлях з найбільшою довжиною $d_c\left(\mu_{sj}^{r=4}\right)$. Це шлях $\mu_{s2345}^{*r=4}$.

Ітерація 5. Будуємо шляхи рангу r=5. Як видно з рис. 3.6, шляхів, що задовольняють умову задачі рангу r=5, немає. Тоді з усіх шляхів μ_{sj}^{*r}, $r=(\overline{1,4})$, $j=(\overline{r,n})$ вибираємо найбільший: $\mu_{s2345}^{*r=4}$, а вектор $\bar{x}^{onm}=(0111100)$, йому відповідний, є оптимальним розв'язком задачі.

Використання стратегії L_1 не завжди дає оптимальний розв'язок задачі. Виникнення ситуації відсікання шляхів за формулою (3.18), що

відповідають оптимальному розв'язку задачі, обумовлено тим, що шляхи з більшим значенням по функціоналу можуть набирати й більшу довжину по вагах обмежень. Отже, на деякому ранзі виникає ситуація, коли через обмеження шляхи наступних рангів у графі $D\Delta$ побудувати неможливо, а шляхи з меншим значенням довжини по вагах функціонала, що мають і меншу довжину по вагах обмежень, відкинуті за стратегію L_1. На їхній основі могли б бути отримані шляхи більшого рангу й за рахунок цього й більшої величини по вагах функціонала.

Позначимо через $\mu_{sj=\gamma}^{**r=q}$ шлях рангу $r=q$ з найбільшою довжиною $d_c\left(\mu_{sj}^{**r}\right)$ серед шляхів $\left\{\mu_{sj}^{*r\neq q}\right\}$ рангу $r<q$, що відповідають локальним екстремумам, обумовленим процедурою A_0 з використанням стратегії L_1. Тоді сформулюємо правило відсікання L_2 для одномірної задачі на основі наступної теореми.

Теорема 3.1. У графі $D\Delta$ не існує шляху, що задовольняє властивість ν рангу $r<q$, для якого виконується нерівність

$$d_c\left(\mu_{sj}^{*r\neq q}\right) < d_c\left(\mu_{sj=\gamma}^{**r=q}\right).$$

Доведення. На кожному етапі формування множин шляхів $m_{sp}^{r=r+1}$ $p=(\overline{r,n})$ процедурою A_0, що використовує правило L_1, у цих множинах будуються шляхи $\mu_{sp}^{r=r+1}$, що задовольняють властивість ν, відповідно до формули (3.18). Припустимо, що в $D\Delta$ існує шлях $\mu_{\bar{s}j=k}^{r\neq q}$, довжина якого $d_c\left(\mu_{\bar{s}j=k}^{r\neq q}\right) < d_c\left(\mu_{sj=\gamma}^{**r=q}\right)$, і він задовольняє властивість ν. Існування такого шляху можливо, якщо існують шляхи $\left\{\mu_{\bar{s}j}^{r\neq q}\right\}$, що задовольняють властивість ν, довжини яких $d_c\left(\mu_{\bar{s}j}^{r\neq q}\right)$ перевищують довжини шляхів $d_c\left(\mu_{sj}^{*r\neq q}\right)$, що відповідають локальним екстремумам, виділеним процедурою A_0 з використанням стратегії L_1. Покажемо, що дане припущення неправильне. Для цього доведемо наступне твердження.

Твердження 3.1. Процедура A_0 з використанням стратегії L_1 дозволяє визначати в графі $D\Delta$ шляхи рангу $r<q$ максимальної довжини по вагах функціонала, що задовольняють властивість ν, від вершини s до усіх інших вершин графа $D\Delta$.

Якщо є правильним твердження 3.1, то це означає, що не існує шляхів $\mu_{\bar{s}j}^{r\neq q}$, що задовольняють нерівність $d_c\left(\mu_{\bar{s}j}^{r\neq q}\right) > d_c\left(\mu_{sj}^{*r\neq q}\right)$, і, отже,

припущення про існування шляху $\mu_{sj=k}^{r<q}$, для якого виконується нерівність $d_c\left(\mu_{sj=k}^{r=q}\right) > d_c\left(\mu_{sj}^{**r=q}\right)$, теж не є правильним, і теорема доведена.

Правомірність твердження 3.1 для множин $m_{sj}^{r=1}$, $m_{sj}^{r=2}$ очевидна, оскільки множини $m_{sj}^{r=1}$ містять по одному шляху рангу r=1, а в множинах $m_{sj}^{r=2}$ містяться всі шляхи $\mu_{sj}^{r=2}$ рангу r=2 графа DΔ. Нехай на основі множин $m_{sj}^{r=2}$, $j=\overline{(r,n)}$ процедурою A_0 з використанням стратегії L_1 сформовані множини $m_{sj}^{r=3}$. Виділимо в $m_{sj}^{r=3}$ шляхи $\mu_{sj}^{*r=3}$ максимальної довжини по вагах функціонала. Припустимо, що в графі $D\Delta$ існує шлях $\mu_{sj}^{*r=3}$, що задовольняє властивість ν, але довший, ніж $\mu_{sj}^{*r=3}$. Відповідно до рекурентних співвідношень (3.18) останнє можливо, якщо в множинах $m_{sj}^{r=2}$ є шляхи $\mu_{sj}^{*r=2}$ довші, ніж шляхи $\mu_{sj}^{*r=2}$. Але це суперечить раніше встановленому факту, що шляхи $\mu_{sj}^{*r=2}$ найдовші шляхи рангу r=2 у графі DΔ. Отже, припущення про існування шляху $\mu_{sj}^{**r=3}$ не є правильним, а отже, твердження 3.1 справедливо й для множин $m_{sj}^{r=3}$.

Припустимо, що воно виконується й для $m_{sj}^{r=k<q}$, і доведемо, що воно є правильним і для множин $m_{sj}^{r=k+1}$. Нехай на базі множин шляхів $m_{sj}^{r=k}$ процедура A_0, що використовує стратегію L_1, побудувала множини шляхів $m_{sj}^{r=k+1}$. Виділимо в множинах $m_{sj}^{r=k+1}$ шляхи максимальної довжини $\mu_{sj}^{*r=k+1}$ і припустимо, що в графі $D\Delta$ існує шлях $\mu_{sj}^{*r=k+1}$, що задовольняє властивість ν, але довше від $\mu_{sj}^{*r=k+1}$. Останнє можливо, якщо існує шлях $\mu_{sj}^{**r=k}$ довший, ніж $\mu_{sj}^{**r=k}$, але задовольняє властивість ν. Але це суперечить первісному припущенню про те, що шляхи $\mu_{sj}^{**r=k}$ найдовші шляхи рангу $r = k$ у графі DΔ, і, отже, допущення про існування шляху $\mu_{sj}^{**r=k+1}$ в $D\Delta$ $r = k + 1$ не є правильним, тобто твердження 3.1 справедливо й для множин $m_{sj}^{r=k+1}$.

Отже, ми показали, що твердження 3.1 є правильним для множин $m_{sj}^{r=1}$, $m_{sj}^{r=2}$, і довели, що воно виконується для $m_{sj}^{r=3}$. Далі було припущено, що воно поширюється й на множини $m_{sj}^{r=k}$, і доведено, що у випадку правильності припущення твердження 3.1 справедливо й для множин рангу $r=k+1$. Отже, на основі принципу повної математичної індукції твердження 3.1 справедливо й для довільних множин $m_{sj}^{r<q}$, тому що при $>q$ множини шляхів $m_{sj}^{r<q}$ порожні, а отже, теорема 3.1 є правильною.

З доведеної теореми 3.1 випливає стратегія відсікань L_2, що дозволяє значення функціонала шляху $\mu_{sj}^{**r=q}$ використовувати як верхню оцінку для відсікання шляхів у множинах $m_{sj}^{r<q}$ графа $D\Delta$. Стратегія L_2 використовується при побудові багатоетапних алгоритмів, тому приклад, що пояснює роботу стратегії L_2, буде розглянутий пізніше.

Не порушуючи стратегій L_1 і L_2, можна ввести додаткове відсікання в множинах шляхів m_{sj}^{r}, що представляє стратегію L_3, засноване на властивості графа $D\Delta$, і полягає у такому. Зі структури графа $D\Delta$ видно, що для кожної його вершини j, що відповідає області Ω_j, вага γ_j дорівнює сумі коефіцієнтів c_j у функціоналі (3.10), що визначається за правилом

$$\gamma_j = c_{j+1} + c_{j+2} + \ldots + c_n,\ \gamma_n = 0;\ j = \overline{(1, n-1)}, \qquad (3.19)$$

являє собою верхню оцінку приросту величини значення локального екстремуму в області Ω_j на усіх наступних рангах. Скорочення кількості формованих шляхів μ_{sj}^{r} у множинах m_{sj}^{r} може бути досягнуто, якщо виконуються умови, обумовлені таким досить очевидним твердженням.

Твердження 3.2. Якщо сума довжини шляху $d_c(\mu_{sj}^{r})$ з підмножини r m_{sj}^{r} у вершину w і ваги γ_w цієї вершини менше від уже отриманої максимальної довжини $d_c(\mu_{sj=p}^{r})$, то шляхи наступного рангу, побудовані на основі μ_{sj}^{r}, не можуть визначати оптимальний розв'язок задачі.

Справедливість твердження 3.2 випливає з того, що процедура A_0 зі стратегією L_1 вибирає глобальний екстремум із усіх локальних, а найбільша довжина за рахунок продовження шляху μ_{sj}^{r} вже менше від одного з локальних екстремумів. Таким чином, перевірка умови

$$d_c\left(\mu_{sp}^{r}\right)+\gamma_p < \max_{\{c_j\}}\left[d_c\left[\mu_{sp}^{*r}\right]\right], \qquad (3.20)$$

де $d_c\left(\mu_{sp}^{r}\right)$ – довжина шляху μ_{sp}^{r} до вершини p рангу r по вагах c_j, дозволяє виключити цей шлях з подальшого аналізу як безперспективний, якщо умова виконана. На цій перевірці заснована стратегія L_3.

Приклад 3.2. Потрібно максимізувати функціонал

$$f(x) = 20x_1 + 18x_2 + 18x_3 + 15x_4 + 10x_5 + 4x_6 + 1x_7$$

при обмеженні $7x_1 + 2x_2 + 5x_3 + 2x_4 + 3x_5 + 1x_6 + 4x_7 \le 12$ у відповідності зі стратегією L_1 і L_3.

Розв'язання. Ілюстрація розв'язання наводиться на рис. 3.7. Тут і далі підкреслення шляху означає те, що тільки на основі цього шляху відбувається побудова множин наступного рангу, інакше шлях виключається з подальшого аналізу. Ліворуч у вигляді стовпця проставлено значення γ_j, j =$(\overline{1,7})$ (формула (3.20)), обумовлене для кожної вершини графа $D\Delta$. Так, наприклад $\gamma_3=c_4+c_5+c_6+c_7=15+10+4+1=30$.

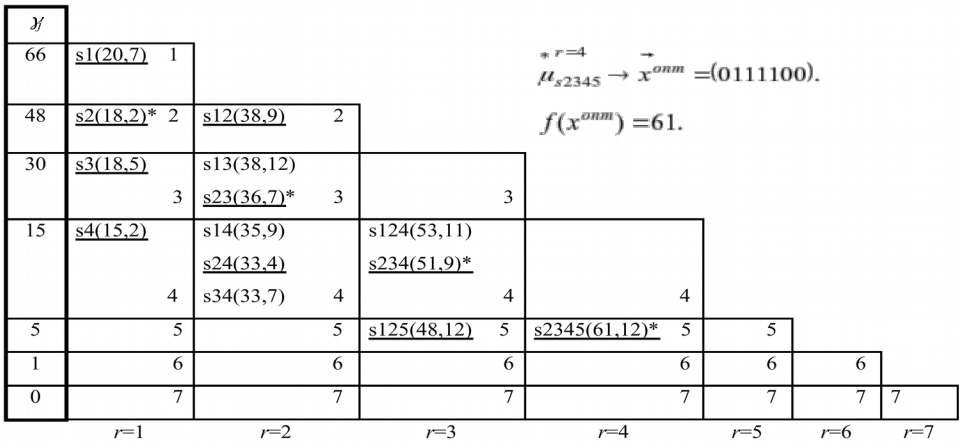


Рис. 3.7. Ілюстрація роботи стратегій L_1 і L_3

Відмінною рисою від прикладу 3.2 є наявність порожніх множин m_{sj}^{r}. Це обумовлено застосуванням відсікань за формулою (3.20). Наприклад, побудуємо шлях $\mu_{s15}^{r=2}$. До цього часу $\max\left[d_c\left[\mu_{s12}^{*r=2}\right]\right]=38$, а $d_c\left(\mu_{s12}^{r=2}\right)=30$. Тоді $d_c\left(\mu_{s15}^{r=2}\right)+\gamma_5=35$. Отже, на основі шляху $\mu_{s15}^{r=2}$ згідно з твердженням

3.2 неможливо побудувати шлях, що перевищує по вагах функціонал значення вже існуючого локального екстремуму: шлях $\mu_{s12}^{*r=2}$.

Як видно з прикладу 3.2, спільне застосування стратегій L_1 і L_3 дозволяє в 3 рази зменшити кількість векторів, що будуються в множинах m_{sj}^r, порівняно з застосуванням однієї стратегії L_1.

У результаті роботи процедури A_0 зі стратегіями L_1, L_3 залишаються множини m_{sj}^r, для яких $r>q$ (у прикладах 3.1 і 3.2 вони є порожніми), а для ефективної фільтрації безперспективних шляхів необхідно мати оцінки значення функціонала в областях Ω_j, $j=\overline{1,n^2/2}$ при $r>q$. Для цього введемо стратегію вибору шляхів із множин L_4, що ґрунтується на процедурі, що дозволяє одержати шляхи максимально можливого рангу r у графі $D\Delta$. Її можна реалізувати, якщо в $D\Delta$ визначати найкоротші шляхи по вагах обмежень на основі процедури A_0 між вершиною s і всіма іншими вершинами графа $D\Delta$, при цьому рекурентне співвідношення (3.17) для одномірної задачі (3.10) – (3.12) набуде такого вигляду:

$$\mu_{sp}^{r=r+1}=\min_{a_{ij}}\left|\mu_{sj}^r\cup(j,p)\right|;\ \mathrm{p}=\overline{(\mathrm{r}+1,\mathrm{n})};\ \ \mathrm{j}=\overline{(\mathrm{r},\mathrm{n})},\ \ \mathrm{i}=1. \qquad (3.21)$$

Неважко побачити, що при цій стратегії формування шляхів у множинах m_{sj}^r залишається справедливим твердження 3.2 і для відсікання шляхів у множинах m_{sj}^r можна використовувати умову (3.20).

Приклад 3.3. Потрібно максимізувати функціонал

$$f(x)=47x_1+45x_2+38x_3+32x_4+22x_5+22x_6+20x_7+18x_8$$

при обмеженні $10x_1+4x_2+10x_3+8x_4+9x_5+5x_6+1x_7+1x_8\le 30$ у відповідності зі стратегією L_3 і L_4 (рис. 3.8).

Розв'язання. З кожної множини m_{sj}^r для подальшого аналізу вибираються шляхи $\mu_{sj}^r\in m_{sj}^r$, що мають мінімальне значення по вазі $d_a\left(\mu_{sj}^r\right)_1$ (на рис. 3.9 підкреслені). Інші – виключаються з подальшого аналізу за формулою (3.21). На ранзі $r=4$ у множині $m_{s6}^{r=4}$ шлях $\mu_{s1246}^{r=4}$, що лежить в основі оптимального, губиться через те, що існує шлях $\mu_{s2456}^{r=4}$ менший по вагах функціонала й менший по вагах обмеження.

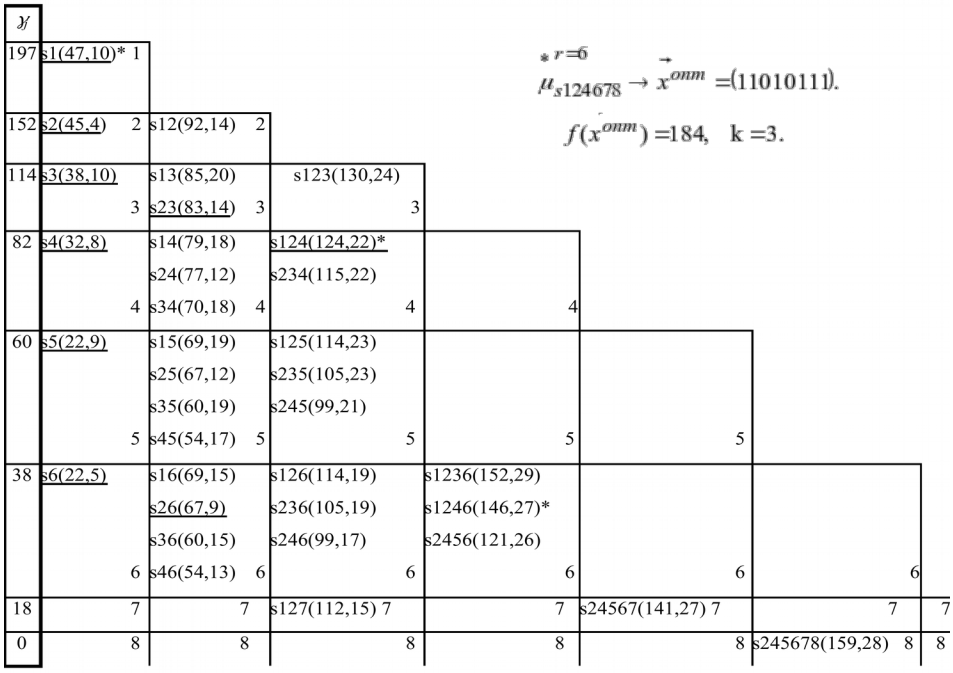


Рис. 3.8. Ілюстрація роботи стратегій L_3 і L_4

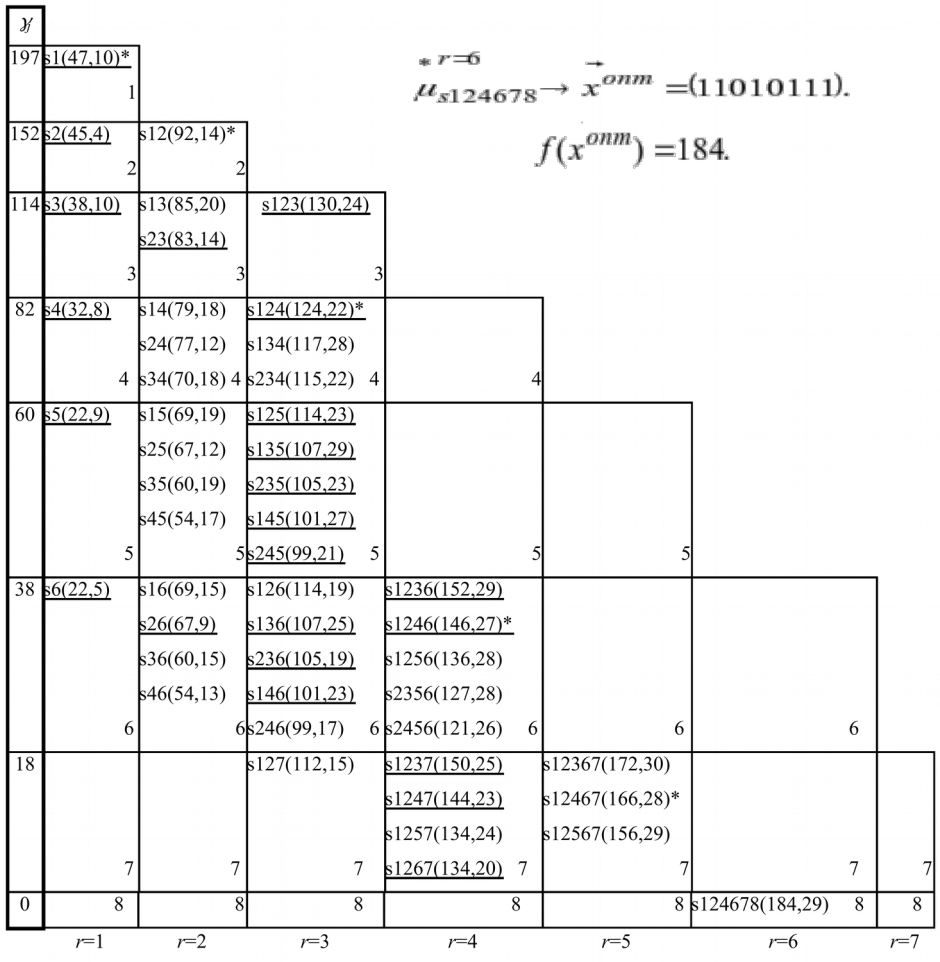


Рис. 3.9. Ілюстрація роботи стратегій L_3 і L_6

Слід зазначити також, що для даного прикладу застосування стратегій L_3 і L_4 не дає оптимального розв'язку задачі. Так, отриманий шлях $\mu^{*r=6}_{s245678}$ не є оптимальним, тому що останній був відкинутий на ранзі $r = 4$ у множині $m^{r=4}_{s6}$ із зазначеної вище причини.

Реалізація процедури A_0 з урахуванням стратегій L_3, L_4 дозволяє сформулювати додаткову стратегію L_5 відсікання безперспективних шляхів у множинах m^r_{sj}, обумовлену наступною теоремою.

Теорема 3.2. Якщо з множини $m^r_{sj}, j = \overline{(r_{\max}, n)}$ максимального рангу r_{max}, побудованого процедурою A_0 зі стратегіями L_3, L_4, вибрати шлях μ^r_{sw} з найбільшим значенням по вагах функціонала c_j, то у випадку повторного розв'язання вихідної задачі досить формувати шляхи першого рангу r=1 у множині

$$m^r_{sp}, p = \overline{(1,k)}, \text{ де } k = w - r_{\max} + 1, \qquad (3.22)$$

а w – номер множини m^r_{sw} зі шляхом максимальної довжини по вагах функціонала, обумовлений при першому розв'язанні задачі.

Доведення. Припустимо, що існує шлях μ^{*r}_{sp} ($p>k$), що досягає значення максимального рангу r_{max}. Оскільки $p>k$, то шлях $\mu^{*r=\kappa}_{sp}$ у графі $D\Delta$ буде проходити по нижніх вершинах (рис. 3.3). Але ваги c_j згідно з формулою (3.12) відсортовані в порядку зменшення й вершинам, що перебувають на більш низьких горизонтальних лінійках, у $D\Delta$ відповідають і менші ваги по функціоналу. Тому на кожному ранзі шлях μ^{*r}_{sp} зможе набрати не більш, ніж шлях μ^r_{sw}, а отже, він і в сумі набере по вагах $\{c_j\}$ довжину не більшу, ніж μ^r_{sw}, що було потрібно довести.

Відповідно до стратегії відсікань L_5 у випадку повторного розв'язання задачі (приклад 3.3) за будь-яким алгоритмом будувати вектори множин з $m^{r=1}_{s4}$ по $m^{r=1}_{s8}$ не потрібно, тому що кінцевою вершиною шляху максимального рангу $\mu^{r=6}_{s256788}$ є вершина w=8. Ранг цього шляху дорівнює $r_{max} = 6$, за формулою (3.22) $k = 3$, а отже, будувати шляхи першого рангу необхідно тільки в $m^{r=1}_{s1}$, $m^{r=1}_{s2}$, $m^{r=1}_{s3}$.

Найбільш важливою стратегією L_6 є стратегія, заснована на понятті виділення коридору в множині m^r_{sj}, що дозволить будувати точні алгоритми розв'язання задачі (3.10) – (3.12). Оскільки нумерація змінних і вершин у графі $D\Delta$ відповідає порядку зменшення коефіцієнтів c_j у функціоналі, а процедура A_0 на кожному етапі формує множини m^r_{sj},

починаючи з індексів $j = (\overline{r,n})$, то шляхи $\mu_{sj}^{r} \in m_{sj}^{r}$ виявляться завжди відсортованими в порядку зменшення довжин по вагах функціонала.

У вигляді наступної теореми сформулюємо правило, що дозволяє в множині m_{sj}^{r} виділити шлях $\overline{\mu}_{sj}^{r}$ з довжиною $d_c\left(\overline{\mu}_{sj}^{r}\right)$, стосовно якого всі шляхи з меншим значенням довжини по вагах функціонала можуть бути виключені з аналізу як неперспективні.

Теорема 3.3. Якщо відсортувати в множині $m_{sj}^{r}, j = (\overline{r,n})$ рангу r вектори в порядку зменшення по вагах функціонала c_j, то шляхи, у яких довжина $d_c\left(\mu_{sj}^{r}\right)$ менше від довжини $d_c(\min_{a_{ij}} \mu_{sj}^{r})$, не можуть визначати оптимальний розв'язок задачі.

Доведення. Припустимо, що на основі шляху μ_{sj}^{*r} із множини m_{sj}^{r}, у якого довжина $d_c(\mu_{sj}^{*r})$ менше, ніж довжина $d_c(\min_{a_{ij}} \mu_{sj}^{r})$ вдалося побудувати оптимальний розв'язок задачі (3.14) – (3.17). Тоді в множини $m_{sp}^{r=r+1}, p = (\overline{r,n})$ наступного рангу повинен потрапити шлях і мінімальний по вагах обмежень, адже він тим більше буде задовольняти властивість ν. Однак довжина по вагах функціонала такого шляху буде більше, ніж в μ_{sj}^{*r} і, отже, припущення про існування μ_{sj}^{*r} не є правильним, а цей шлях можна виключити з подальшого аналізу, що й було потрібно довести.

Приклад 3.4. Потрібно максимізувати функціонал

$$f(x) = 47x_1 + 45x_2 + 38x_3 + 32x_4 + 22x_5 + 22x_6 + 20x_7 + 18x_8$$

при обмеженні $10x_1 + 4x_2 + 10x_3 + 8x_4 + 9x_5 + 5x_6 + 1x_7 + 1x_8 \leq 30$ у відповідності зі стратегією L_3 і L_6.

Розв'язання. Побудова шляхів наступного рангу буде здійснюватися на основі шляхів попереднього рангу. Всі шляхи, на основі яких і відбувається така побудова, виділені підкресленням (рис. 3.9). Для подальшого аналізу беруться тільки два шляхи $\mu_{s125}^{r=3}$ і $\mu_{s235}^{r=3}$, які підкреслені. Інші виключаються з множини $m_{s5}^{r=3}$. Аналогічно будуються множини шляхів усіх рангів.

Так, наприклад, у множині $m_{s5}^{r=2}$ шляхи $\mu_{s35}^{r=2}$, $\mu_{s45}^{r=2}$ виключаються з подальшого розгляду, тому що згідно з теоремою 3.2 існує шлях $\mu_{s25}^{r=2}$ у вершину 5, більший по вагах функціонала й менший по вагах обмежень. На відміну від прикладу 3.3, застосування стратегій L_3 і L_6 дозволяє точно розв'язати дану (і будь-яку іншу) задачу.

Теорема 3.4 визначає поняття коридору.

Визначення. Під *одномірним коридором* із множини m_{sj}^{r} в множину $m_{sp}^{r=r+1}$ будемо розуміти сукупність шляхів μ_{sj}^{r}, що перебувають між верхньою границею множини m_{sj}^{r} і його нижньою границею, що задовольняють властивість ν у множині $m_{sp}^{r=r+1}$. Верхня границя визначається шляхом з максимальної $d_c\left(\mu_{sj}^{r}\right)$, а нижня – теоремою 3.3.

Виходячи з поняття коридору можна запропонувати таку стратегію L_7, що являє собою ще одну стратегію вибору шляхів, що r полягає в тім, щоб із множини m_{sj}^{r} рангу r у множині $m_{sp}^{r=r+1}$, $p=(\overline{r,n})$ наступного рангу вибирати шляхи, що задовольняють властивість ν і є максимальними по вагах функціонала c_j і мінімальними по вагах обмежень a_{1j}, що відповідає рекурентним співвідношенням (3.18), (3.21).

Приклад 3.5. Потрібно максимізувати функціонал

$$f(x) = 50x_1 + 50x_2 + 45x_3 + 41x_4 + 37x_5 + 36x_6 + 31x_7 + 27x_8$$

при обмеженні $9x_1 + 7x_2 + 6x_3 + 9x_4 + 6x_5 + 6x_6 + 5x_7 + 1x_8 \le 27$ у відповідності зі стратегією L_3 і L_7 (рис. 3.10).

Розв'язання. Даний приклад розкриває поняття одномірного коридору, тобто під коридором у множині $m_{s5}^{r=3}$ розуміється підмножина векторів, що складається зі шляхів $\mu_{s125}^{r=3}$, $\mu_{s135}^{r=3}$, $\mu_{s235}^{r=3}$, а у відповідності зі стратегією L_7 для подальшого аналізу беруться тільки два шляхи $\mu_{s125}^{r=3}$ і $\mu_{s235}^{r=3}$, які підкреслені. Інші виключаються з множини $m_{s5}^{r=3}$. Аналогічно будуються множини шляхів усіх рангів.

Розглянемо тепер правила фільтрації безперспективних шляхів усередині виділеного коридору, починаючи зі стратегії L_8 , заснованої на наступному твердженні.

Твердження 3.3. Якщо в коридорі існують два шляхи $\mu_{sj}^{*r} \in m_{sj}^{r}$ й $\mu_{sj}^{**r} \in m_{sj}^{r}$, для яких $d_c(m_{sj}^{*r}) > d_c(m_{sj}^{**r})$ $d_a(m_{sj}^{*r})_1 \le d_a(m_{sj}^{**r})$, то вектор x, що відповідає шляху μ_{sj}^{*r}, не може належати оптимальному розв'язку задачі (3.10) – (3.12). Покажемо справедливість твердження 3.3. Для цього припустимо, що на основі шляху μ_{sj}^{**r} можна побудувати оптимальний розв'язок.

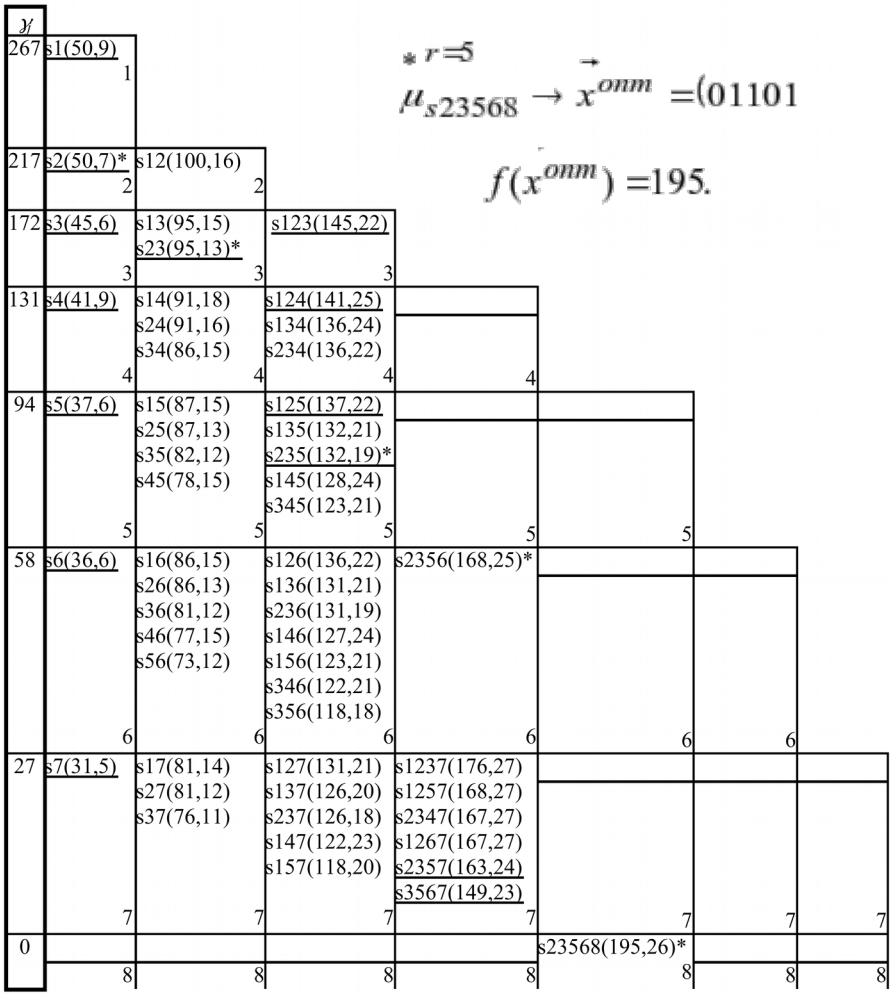


Рис. 3.10. Ілюстрація роботи стратегій L_3 і L_7

Останнє означає, що на наступних рангах шлях μ_{sj}^{**r} набере більшого значення по вагах функціонала, ніж шлях μ_{sj}^{*r}, тобто $d_c(m_{sj}^{*r}) < d_c(m_{sj}^{**r})$. Але шлях μ_{sj}^{*r} може бути побудований у тій множині m_{sj}^{r}, що й шлях μ_{sj}^{**r}, що задовольняє властивість ν, але має довжину $d_a(m_{sj}^{*r}) \le d_a(m_{sj}^{**r})$. При цьому по вагах функціонала до цього моменту шлях μ_{sj}^{*r} має довжину більшу, ніж шлях μ_{sj}^{**r}. Отже, кожний з них набере ту саму величину по вагах c_j, але $d_c(m_{sj}^{*r}) > d_c(m_{sj}^{**r})$. Отже, припущення про те, що на основі шляху μ_{sj}^{**r} можна побудувати оптимальний розв'язок не є привильним, а твердження 3.3 справедливо.

Приклад 3.6. Потрібно максимізувати функціонал

$$f(x) = 47x_1 + 45x_2 + 38x_3 + 32x_4 + 22x_5 + 22x_6 + 20x_7 + 18x_8$$

при обмеженні $10x_1 + 4x_2 + 10x_3 + 8x_4 + 9x_5 + 5x_6 + 1x_7 + 1x_8 \leq 30$ у відповідності зі стратегією L_3 і L_8 (рис. 3.11).

Розв'язання. Розглянемо застосування стратегії L_8 до однієї з множин графа $D\Delta$ – множини $m_{s5}^{r=3}$. У результаті в ній залишаються лише два шляхи $\mu_{s125}^{r=3}$, $\mu_{s245}^{r=3}$, що відповідають двом векторам: (11001000) і (01011000), а без цього правила їх було б п'ять (див. приклад 3.4). Аналогічний виграш і для інших множин.

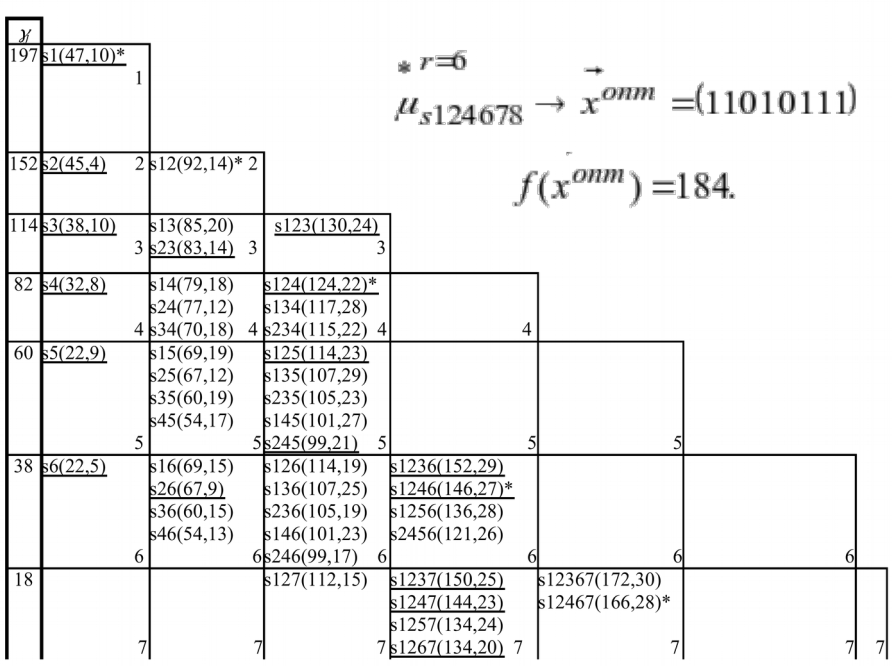


Рис. 3.11. Ілюстрація застосування стратегій L_3 і L_8

Далі розглянемо стратегію L_9 фільтрації розв'язків у коридорі на основі *каліброваних векторів.* Для цього введемо поняття каліброваного вектора $y_j, j = (\overline{1,n})$ для кожної вершини j графа $D\Delta$. Множина $\{y_j\}$ утворить калібровану шкалу, що дає верхню оцінку $\overline{r}_{\hat{a}}$ для будь-якого вектора x, що відповідає шляху μ_{sj}^{r} і дозволяє визначити, на скільки рангів можна ще продовжити шлях μ_{sj}^{r}, що задовольняє властивість ν. Побудову каліброваних векторів y_j почнемо з першої вершини графа DΔ, спочатку відсортувавши коефіцієнти $\{a_{1j}\}$ у порядку зростання. Утвориться вектор $a_{1q}^{'}$, $q = (\overline{1,n})$, у якого $a_{11}^{'} \leq a_{12}^{'} \leq ... \leq a_{1n}^{'}$. У цьому векторі $a_{1q}^{'}$ викреслюємо $a_{1q}^{'} = a_{11}$, після чого формуємо компоненти каліброваного вектора y_1 за правилом

$$y_{1k} = a_{1k} + y_{1(k-1)}; \quad k = (\overline{1,n-1}); \quad y_{10} = 0. \tag{3.23}$$

Далі в a'_{1q} викреслюємо $a'_{1q} = a_{12}$, після чого аналогічно правилу (3.23) продовжуємо формувати компоненти вектора y_2, y_3 і т. д. На j–му кроці викреслюємо в a'_{1q} елемент $a'_{1q} = a_{1j}$, тоді

$$\begin{aligned} &y_{jk} = a_{jk} + y_{j(k-1)}; \quad k = \overline{(1, n-j)}; \\ &y_{10} = 0; \quad y_{n0} = b_1; \quad j = \overline{(1, n-1)}. \end{aligned} \tag{3.24}$$

Приклад 3.7. Потрібно максимізувати функціонал

$$f(x) = 20x_1 + 15x_2 + 12x_3 + 8x_4 + 2x_5 \tag{3.25}$$

при обмеженні $10x_1 + 3x_2 + 7x_3 + 4x_4 + 2x_5 \le 10$……(3.26)

Для задачі (3.25)–(3.26) вектор $\bar{a}'_{1q} = \{2,3,4,7,10\}$. З нього видаляємо $a'_{15} = a_{11} = 10$, тоді $a'_{1q} = \{2,3,4,7\}$, а $y_1 = \{0,2,5,9,16\}$. Далі викреслюємо $a'_{1q} = a_{12} = 3$. Вектор $a'_{1q} = \{2,4,7\}$, а калібрований вектор $y_2 = \{0,2,6,13\}$. Аналогічним образом за допомогою формули (3.24) будуються калібровані вектори для вершин 3, 4, 5: $y_3 = \{0,2,6\}$; $y_4 = \{0,2\}$; $y_5 = \{0\}$. Сформулюємо правила, за якими за допомогою каліброваного вектора y_j для вершин $j = \overline{(1,n)}$ можна визначити шляхи μ^r_{sj} в графі $D\Delta$ і верхню оцінку $\hat{r}_в$ максимального значення рангу, що продовжує r шлях μ^r_{sj} у графі DΔ (тобто яку кількість одиниць може додатися у вектор x).

Правило K_1

а) обчислюємо для шляху μ^r_{sj} величину $\Delta d = b_1 - d_a(\mu^r_{sj})$;

б) у каліброваному векторі y_j для вершини j знаходимо номер k, $k = \overline{(0, n-j)}$ елемента y_{jk}, починаючи з якого $y_{jk} > \Delta d$ (якщо $\forall\, y_{jk} < \Delta d$, то $k = n - j + 1$);

в) визначаємо значення $\hat{r}_в$ для шляху μ^r_{sj} в графі DΔ: $\hat{r}_а = k - 1$.

У прикладі (3.25) – (3.26) шлях $\mu^{r=1}_{s2}$ має $d_a(\mu^{r=1}_{s2}) = 3$. Шляху $\mu^{r=1}_{s2}$ відповідає вектор $x = (01000)$. Визначимо $\hat{r}_в$ $(\mu^{r=1}_{s2})$ за правилом K_1.

а) $\Delta d = b_1 - d_a(\mu^{r=1}_{s2}) = 10 - 3 = 7$;

б) в y_2 елемент $y_{23} > \Delta d$ (13 > 7), отже $k = 3$;

в) значення $\hat{r}_а(\mu^{r=1}_{s2}) = k - 1 = 3 - 1 = 2$.

Отже, шлях в $\mu_{s2}^{r=1}$ у графі DΔ можна продовжити, задовольняючи обмеження, не більш, ніж на два ранги. Справедливість застосування правила K_1 для визначення $\hat{r}_в$ підтверджується наступною теоремою.

Теорема 3.5. Якщо шлях μ_{sj}^{r} у вершину j рангу r у графі DΔ має довжину по вагах обмежень $d_a(\mu_{sj}^{r}) = d$, то верхня оцінка $\hat{r}_в$, побудована за правилом K_1, визначає максимальну кількість рангів r, на яке може бути продовжений шлях μ_{sj}^{r} у графі DΔ.

Доведення. Припустимо, що для шляху μ_{sj}^{r} з вершини j знайшлося таке його продовження $\mu_{sj}^{r'}$, ранг якого $r' > \hat{r}_в + 1$. Але відповідно до пункту а) правила K_1 $\Delta d'$ для $\mu_{sj}^{r'}$ буде більше Δd для шляху $\mu_{sj}^{r+\hat{r}_в}$, що неможливо, тому що b_1 фіКЗовано й обидва шляхи є продовженням шляху μ_{sj}^{r} з довжиною $d_a(\mu_{sj}^{r}) = d$. Отже, $d' = \Delta d$ і припущення про існування $\mu_{sj}^{r'}$ не є правильним і теорема доведена.

Ми не враховували величину коефіцієнтів c_j при функціоналі, а прогноз здійснювали тільки на основі коефіцієнтів a_{ij}. Поставимо у відповідність кожному вектору y_j вектор z_j, елементи якого є верхніми оцінками $\hat{z}_в$ довжини шляхи μ_{sj}^{r} по вагах функціонала c_j. Компоненти вектора z_j для вершини j формуються відповідно до правила K_2.

Правило K_2

Нехай перша компонента вектора z_j вершини j дорівнює $\hat{z}_{j1}^{в} = c_{j+1}$, друга – $\hat{z}_{j2}^{в} = c_{j+2} + \hat{z}_{j1}^{в}$; третя $\hat{z}_{j2}^{в} = c_{j+3} + \hat{z}_{j2}^{в}$ і т. д., k–та компонента –

$$\hat{z}_{jk}^{в} = c_{j+k} + \hat{z}_{j(k-1)}^{в};\ k = \overline{(1, n-j)};\ \hat{z}_{j0}^{в} = 0;\ \hat{z}_{n0}^{в} = 0;\ j = \overline{(1, n-1)}. \qquad (3.27)$$

Тоді за формулами (3.27) побудуємо z_j для задачі (3.25) – (3.26). У вершині $j = 1$ елементи вектора z_1 відповідно до правила K_2 будуть

$$\hat{z}_{10}^{а} = 0;\ \hat{z}_{11}^{а} = c_{j+1} = c_2 = 15;\ \hat{z}_{12}^{а} = c_3 + \hat{z}_{11}^{а} = 12 + 15 = 27;$$
$$\hat{z}_{13}^{а} = c_4 + \hat{z}_{12}^{а} = 8 + 27 = 35;\ \hat{z}_{14}^{а} = c_5 + \hat{z}_{13}^{а} = 2 + 35 = 37.$$

Для вершини j=2 згідно з формулою (3.27) маємо

$$\hat{z}_{20}^{а} = 0;\ \hat{z}_{21}^{а} = c_3 = 12;\ \hat{z}_{22}^{а} = c_4 + \hat{z}_{21}^{а} = 8 + 12 = 20;\ \hat{z}_{23}^{а} = c_5 + \hat{z}_{22}^{а} = 2 + 20 = 22.$$

Аналогічно будуються вектори для інших вершин графа DΔ. Таким чином, для прикладу (3.19)–(3.20) калібровані оцінні вектора для вершин $j = \overline{(1,n)}$ мають вигляд

$$
\begin{array}{llll}
j = 1; & y_1 = \{0,2,5,9,16\}; & z_1 = \{0,15,27,35,37\}; & \gamma_1 = 37; \\
j = 2; & y_2 = \{0,2,6,13\}; & z_2 = \{0,12,20,22\}; & \gamma_2 = 22; \\
j = 3; & y_3 = \{0,2,6\}; & z_3 = \{0,8,10\}; & \gamma_3 = 10; \\
j = 4; & y_4 = \{0,2\}; & z_4 = \{0,2\}; & \gamma_4 = 2; \\
j = 5; & y_5 = \{0\}; & z_5 = \{0\}; & \gamma_5 = 0.
\end{array}
$$

Неважко побачити, що у векторах z_j останні елементи збігаються з γ_j – вагою (3.19), що може набрати вагу шляху μ_{sj}^r з вершини j на усіх наступних рангах при задоволенні ними властивості ν. Як випливає з формули (3.20), на основі рангового підходу можна більш точно визначати верхню оцінку $\hat{z}_в = f(\hat{r}_в)$ за рахунок того, що шлях μ_{sj}^r не завжди може бути продовжений на всі ранги, як такі, що не задовольняють властивість ν. Інакше кажучи, на підставі рівності

$$\gamma_j = \hat{z}^{в}_{j\hat{r}_в}\left(\mu_{sj}^r\right) \tag{3.28}$$

можна збільшити ефективність фільтрації безперспективних шляхів, заміняючи у формулі (3.20) γ_p на формулу (3.28), що дає стратегія L_9 на основі такої нерівності:

$$d_c\left(\mu_{sp}^r\right) + \hat{z}^{в}_{p_{\hat{r}_в}}\left(\mu_{sp}^r\right) < \max_{c_j}\left|d_c\left(\mu_{sp}^{*r}\right)\right|. \tag{3.29}$$

Перевірка нерівності (3.29) дозволить викреслити шлях μ_{sp}^r з подальшого аналізу як неперспективний, тому що ми заздалегідь знаємо, що більш ніж на $\hat{r}_в$ рангів шлях μ_{sp}^r не може бути продовжений.

Приклад 3.8. Потрібно максимізувати функціонал

$$f(x) = 47x_1 + 45x_2 + 38x_3 + 32x_4 + 22x_5 + 22x_6 + 20x_7 + 18x_8$$

при обмеженні $10x_1 + 4x_2 + 10x_3 + 8x_4 + 9x_5 + 5x_6 + 1x_7 + 1x_8 \leq 30$ у відповідності зі стратегіями L_3, L_8 і L_9 (рис. 3.12).

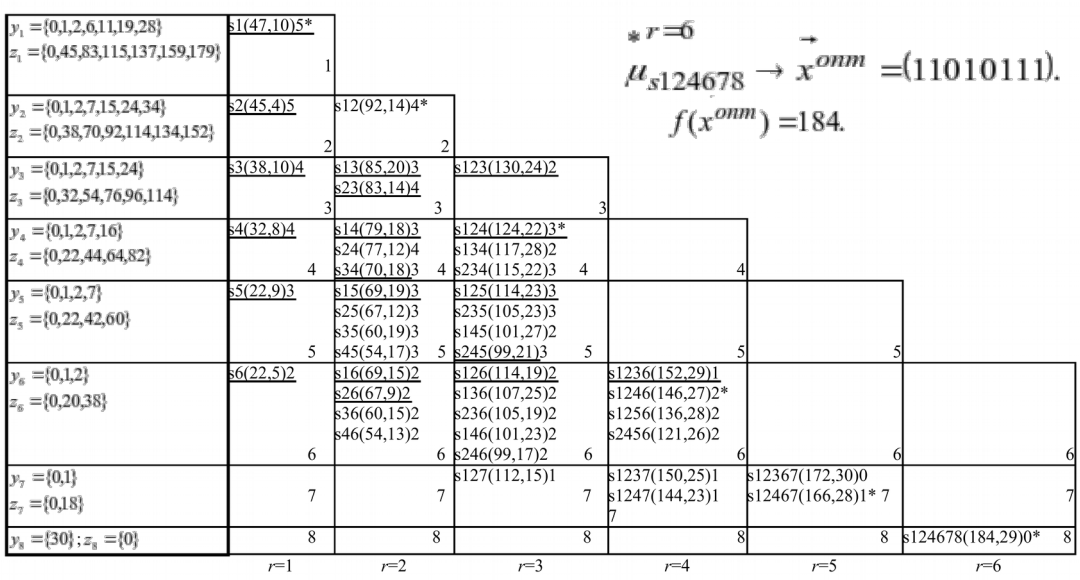


Рис. 3.12. Ілюстрація роботи стратегій L_3, L_8 і L_9

Розв'язання. На початку кожної вершини j графа DΔ зазначені калібровані вектори y_j й $\overset{\sqcup}{z}_j, j = \overline{(1,n)}$. За дужками кожного шляху μ_{sj}^r показана величина $\hat{r}_в$. Так, для шляху $\mu_{s14}^{r=2}$ оцінка дорівнює $\hat{r}_в$=3. Додаткове використання правила K_2 сприяло ефективному відсіканню безперспективних варіантів розв'язання.

Дотепер розглядалася множина $\{L_w\}$ стратегій відсікань для одномірної задачі ЦЛП із БЗ. Сформулюємо стратегії відсікань для m–мірних задач. У випадку розв'язання m-мірних задач найбільш простою стратегією вибору є стратегія L_{10}, заснована на рекурентному співвідношенні (3.18). Принцип формування шляхів у множинах ідентичний стратегії L_1, за винятком необхідності виконувати m раз перевірку виразу (3.8).

Приклад 3.9. Потрібно максимізувати функціонал

$$f(x) = 16x_1 + 15x_2 + 12x_3 + 10x_4 + 8x_5 + 5x_6 + 2x_7$$

при обмеженнях $4x_1 + 3x_2 + 1x_3 + 2_4 + 3_5 + 2x_6 + 1x_7 \le 9$,
$3x_1 + 1x_2 + 2x_3 + 1x_4 + 1x_5 + 3x_6 + 1x_7 \le 9$ у відповідності зі стратегіями L_3 і L_{10} (рис. 3.13).

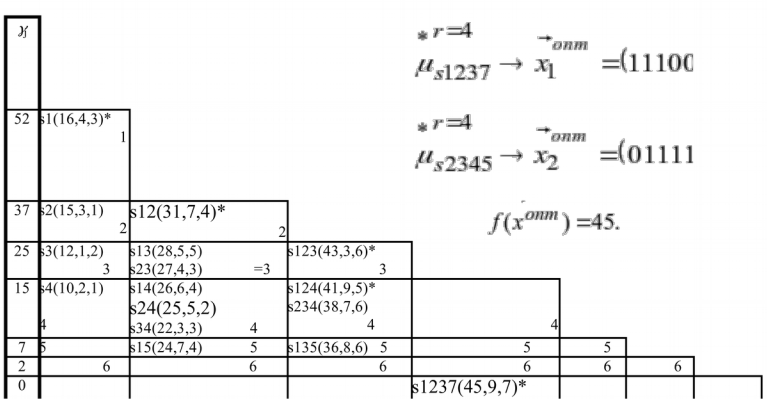


Рис. 3.13. Ілюстрація роботи стратегій L_3 і L_{10}

Розв'язання. Побудова множин m_{sj}^r здійснюється точно так, як і в одномірній задачі. У цьому прикладі два оптимальних розв'язки: шляхи $\mu_{s1237}^{r=4}$ й $\mu_{s2345}^{r=4}$. Однак другий губиться через наявність у множині $m_{s3}^{r=2}$ шляху $\mu_{s13}^{r=2}$, що домінує над $\mu_{s23}^{r=2}$. Поєднання стратегії L_7 зі стратегією L_3 для випадку m–мірної задачі утворює правило вибору L_{11}, що полягає в тім, щоб із множини m_{sj}^r в множину $m_{sp}^{r=r+1}, p = (\overline{r,n})$ вибирати шляхи, що задовольняють властивість ν і є максимальними по вагах функціонала c_j і мінімальними по кожній *i*–й вазі обмежень $a_{ij}, i = (\overline{1,m})$.

Перша стратегія L_7 відповідає рекурентному співвідношенню (3.18), а друга L_3 описується таким співвідношенням, заснованним на виразі (3.21):

$$\mu_{sp}^{r=r+1} = \min_{a_{ij}} \left| \mu_{sj}^r \cup (j,p) \right|; \; j = (\overline{r,n}); \quad p = (\overline{j+1,n}); \; i = (\overline{1,m}). \tag{3.30}$$

Приклад 3.10. Потрібно максимізувати функціонал

$$f(x) = 29x_1 + 28x_2 + 26x_3 + 25x_4 + 20x_5 + 17x_6 + 15x_7 + 10x_8$$

при обмеженнях $1x_1 + 4x_2 + 2x_3 + 3x_4 + 4x_5 + 1x_6 + 2x_7 + 2x_8 \le 9$,
$2x_1 + 2x_2 + 1x_3 + 2x_4 + 4x_5 + 2x_6 + 1x_7 + 4x_8 \le 9$ у відповідності зі стратегіями L_3 і L_{11} (рис. 3.14).

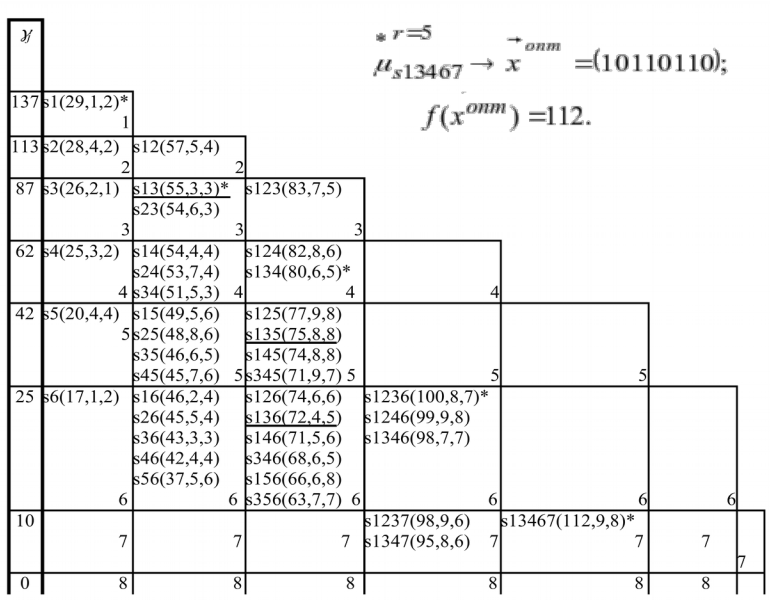


Рис. 3.14. Ілюстрація роботи стратегій L_3 і L_{11}

Розв'язання. Побудова множин шляхів здійснюється відповідно до принципу оптимізації по напрямку (3.17). У множинах m_{sj}^r виділені підкресленням шляхи μ_{sj}^r, що є мінімальними по кожному з обмежень (при рівності *i*–го обмеження вибирається шлях з найбільшою вагою по функціоналу) і максимальними по вазі функціонала. Збільшення кількості обмежень призводить до невизначеності в оцінці верхньої границі по вагах функціонала $\hat{z}_{j\hat{r}_в}^{в}$, яку може набрати шлях μ_{sj}^r, і верхньої оцінки кількості рангів $\hat{r}_в$, на яку може бути продовжений цей шлях. Щоб уникнути цієї невизначеності, для кожної вершини $j = (\overline{1,n})$ будемо будувати не один, а m каліброваних векторів y_j^i, $i = (\overline{1,m})$ за правилом

$$\begin{aligned} &y_{jk}^i = a_{jk}^i + y_{j(k-1)}^i; \quad \mathrm{k} = (\overline{1, \mathrm{n-j}}); \quad \mathrm{y}_{j0}^i = 0; \\ &\mathrm{y}_{n0}^i = b_i; \quad \mathrm{j} = (\overline{1, \mathrm{n-1}}). \end{aligned} \tag{3.31}$$

У зв'язку з цим правило K_1 визначення $\hat{r}_в$ для шляху μ_{sj}^r в багатомірній задачі набуває такого вигляду (правило K_3).

Правило K_3

а) нехай $i = 1$;

б) при поточному значенні *i* обчислюємо для шляху μ_{sj}^r величину $\Delta d = b_i - d_a(\mu_{sj}^r)_i$;

в) у каліброваному векторі $\bar{y}^i_j$ вершини j знаходимо номер k–го $k=(\overline{0,n-j})$ елемента y^i_{jk} вектора $\bar{y}^i_j$, починаючи з якого $y^i_{jk} > \Delta$d (якщо $\forall$ $y^i_{jk} < \Delta$d, то $k=n-j+1$);

г) для шляху μ^r_{sj} за i–м обмеженням у графі DΔ: $\hat{r}^i_в = $k - 1;

д) збільшуємо значення i на 1 і переходимо до пункту б) цього правила у випадку i ≤ m або до наступного пункту у випадку $i > m$;

е) серед усіх $\hat{r}^i_в$, j $=(\overline{1,\text{m}})$ вибираємо найменший.

Таким чином, відповідно до правила K_3 верхня оцінка $\hat{r}_в$ для шляху μ^r_{sj} в j–й вершині графа $D\Delta$ при m обмеженнях буде

$$\hat{r}_в = \min_{\text{i}} \left| \hat{r}^i_в \right|, \quad \text{i} = (\overline{1,\text{m}}). \tag{3.32}$$

Неважко показати, що при реалізації правила K_3 справедливість теореми 3.4 не порушується. Так, якщо за одним з обмежень ми зможемо пройти на r_1 рангів, а за іншим на r_2 ранги й при цьому $r_1 > r_2$, то на ранг більший, ніж r_2, неможливо побудувати шлях, що задовольняє властивість ν, оскільки r_2 – верхня оцінка за даним обмеженням.

Правило K_2 для визначення верхньої оцінки по вагах функціонала в m–мірних задачах не змінюється.

Уведемо за аналогією з поняттям одномірного коридору поняття m-мірного коридору в множині m^r_{sj}.

Позначимо підмножину векторів, що повинна залишитися в множині m^r_{sp} після фільтрації через m^{rk}_{sp}. Нагадаємо, що формування множин шляхів у m^r_{sp} здійснюється завжди таким чином, що довжина по вагах функціонала першого з них більше довжини по вагах функціонала другого й т. д., тобто $d_c(\mu_k) \geq d_c(\mu_k) \geq ... \geq d_c(\mu_k)$.

Визначення. Під виділенням m–мірного коридору будемо розуміти процес відсікання k–го шляху μ_k у множині m^r_{sp}, якому відповідає m–мірний вектор $(d_a(\mu_k)_1, d_a(\mu_k)_2, ..., d_a(\mu_k)_m)$, сформований за правилом K_4.

Правило K_4

Якщо $\left(\forall\ i \in (\overline{1,m})\right) d_a(\mu_v)_1 \leq d_a(\mu_k)_1$, при цьому $=(\overline{1,\text{k}-1})$, то вектор $(d_a(\mu_k)_1, d_a(\mu_k)_2, ..., d_a(\mu_k)_m)$ і відповідний йому шлях μ_k можна видалити з подальшого аналізу, інакше $\mu_k \in m^{r=r+1}_{sp}$

Справедливість відсікання за правилом K_4 випливає з наступного досить очевидного твердження.

Твердження 3.4. Шляхи $\mu_{sp}^{r} \in \mathrm{m}_{\mathrm{sp}}^{\mathrm{r}}$, *що не потрапили в* m_{sp}^{rk} , *не можуть визначати оптимальний розв'язок задачі* (3.1)–(3.3).

Доведення. Нехай деякий шлях $\overset{*}{\mu}{}_{sp}^{r} \notin \mathrm{m}_{\mathrm{sp}}^{\mathrm{r}}$ є оптимальним розв'язком задачі. Тоді відповідно до правила K_4 в m_{sp}^{rk} існує шлях $\overset{**}{\mu}{}_{sp}^{r}$, більший по вагах функціонала й менший по усіх вагах обмежень. Отже, шлях $\overset{**}{\mu}{}_{sp}^{r}$ набере на наступних рангах і більшого значення по вагах функціонала при задоволенні властивості ν, тобто $d_c(\mu_k^{**r}) > d_c(\mu_k^{*r})$. Ми дійшли протиріччя, а отже, наше припущення не є правильним, і твердження 3.4 є справедливим.

Позначимо потужність множини $\left|m_{sj}^{r}\right|$ через Ξ. До відсікання $\left|m_{sj}^{rk}\right| = \oslash$. Тоді процедуру формування m–мірного коридору можна представити у вигляді послідовної перевірки векторів $(1, \Xi)$ із множини m_{sj}^{r} на можливість відсікання за правилом K_4. Вектори, що залишилися, утворять підмножину m_{sj}^{rk} . Цю процедуру будемо називати стратегією L_{12}, що полягає у виділенні m–мірного коридору усередині кожної множини m_{sj}^{r} рангу r, коли в неї побудовані всі шляхи з вищих вершин графа $D\Delta$.

Приклад 3.11. Потрібно максимізувати функціонал

$$f(x) = 17x_1 + 15x_2 + 13x_3 + 8x_4 + 7x_5 + 4x_6 + 4x_7$$

при обмеженнях $2x_1 + 4x_2 + 6x_3 + 9x_4 + 9x_5 + 2x_6 + 1x_7 \le 23$;
$4x_1 + 6x_2 + 9x_3 + 1x_4 + 8x_5 + 10x_6 + 6x_7 \le 33$ у відповідності зі стратегіями L_9 і L_{12} (рис. 3.15).

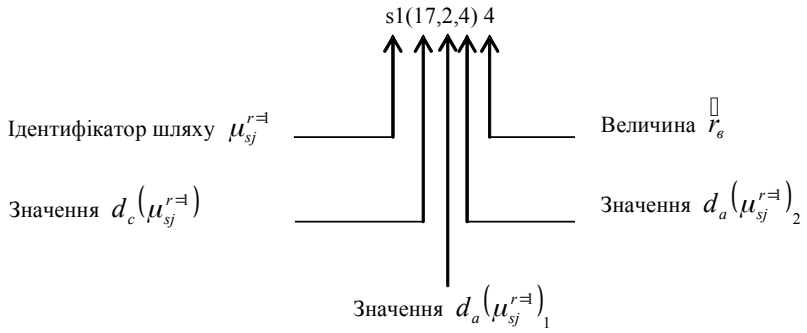


Рис. 3.15

Розв'язання. Побудову шляхів здійснюємо на основі принципу оптимізації по напрямку й за узагальненою процедурою A_0. Кожний шлях μ_{sj}^{r} записується виразом.

Пояснимо порядок роботи правил K_1 і K_3 на прикладі шляху $\mu_{s1}^{r=1}$. Визначимо $\hat{r}_в$. Для цього:

1. Нехай i=1.

2. При поточному значенні i обчислюємо Δd для шляху $\mu_{s1}^{r=1}$:

$$\Delta d = b_i - d_a\left(\mu_{s1}^{r=1}\right)_1 = 23 - 2 = 21.$$

3. У каліброваному векторі y_1^1 вершини 1 знаходимо номер k–го елемента y_{1k}^1 вектора y_j^i, починаючи з якого $y_{1k}^1 > \Delta d \to k = 5$.

4. Для шляху $\mu_{s1}^{r=1}$ за першим обмеженням: $\hat{r}_в^i = k - 1 = 5 - 1 = 4$.

5. Збільшуємо i на 1, тобто i = 2.

6. При поточному значенні i обчислюємо для шляху $\mu_{s1}^{r=1}$ величину

$$\Delta d = b_i - d_a\left(\mu_{s1}^{r=1}\right)_2 = 33 - 4 = 29.$$

7. У каліброваному векторі y_1^2 вершини 1 знаходимо номер k–го елемента y_{1k}^2 вектора y_1^2, починаючи з якого $y_{1k}^2 > \Delta d \to k = 5$.

8. Для шляху $\mu_{s1}^{r=1}$ за другим обмеженням: $\hat{r}_в^i = k - 1 = 5 - 1 = 4$.

9. Серед усіх $\hat{r}_в^i$, $j = \overline{(1,2)}$ вибираємо найменший, тобто $\hat{r}_в = 4$.

Маючи значення $r_â$, за співвідношенням (3.29) перевіряємо на можливість відсікання за стратегією L_9.

Після формування множин усього рангу виробляється виділення m-мірного коридору. Наприклад, на ранзі $r = 2$ у множині $m_{s3}^{r=2}$ шлях $\mu_{s23}^{r=2}$ не потрапляє в коридор, тому що шлях $\mu_{s13}^{r=2} \in m_{s3}^{r=2}$ має більшу довжину по вагах функціонала й меншу довжину по вагах усіх обмежень. У вигляді підкреслення виділено шляхи, що потрапили в коридор.

Однак більш ефективною є стратегія відсікання L_{13}, при якій виділення m-мірного коридору виробляється на всьому ярусі r, після того як на ньому сформуються всі шляхи, що задовольняють у кожній вершині властивість ν. Тоді здійснюється перевірка: чи є для довільного шляху $\mu_{sp}^{r} \in m_{sp}^{r}$ рангу r шлях μ_{sj}^{r}, що належить одній з вищих множин графа $D\Delta$ над m_{sp}^{r}, що має більшу довжину по вагах функціонала й не меншу довжину по вагах усіх m обмежень. Якщо такий шлях μ_{sj}^{r} є, то відповідно до твердження 3.4 шлях μ_{sp}^{r} можна видалити з розгляду як безперспективний.

Приклад 3.12. Потрібно максимізувати функціонал

$$f(x) = 17x_1 + 15x_2 + 13x_3 + 8x_4 + 7x_5 + 4x_6 + 4x_7$$

при обмеженнях $2x_1 + 4x_2 + 6x_3 + 9x_4 + 9x_5 + 2x_6 + 1x_7 \le 23$;
$4x_1 + 6x_2 + 9x_3 + 1x_4 + 8x_5 + 10x_6 + 6x_7 \le 33$ у відповідності зі стратегіями L_9 і L_{13} (рис. 3.16).

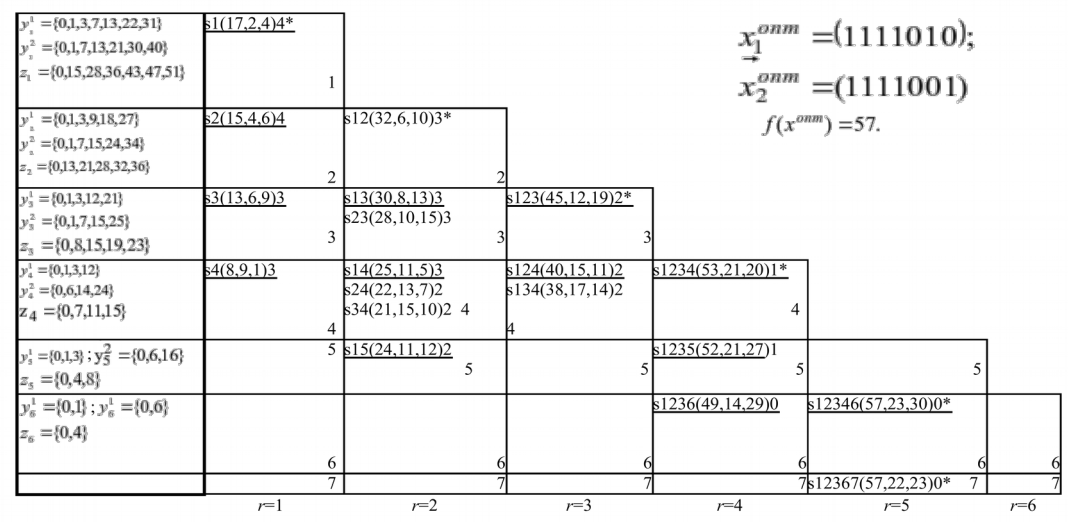


Рис. 3.16. Ілюстрація роботи стратегій L_9 і L_{12}

Розв'язання. Процес розв'язання ідентичний описаному процесу в прикладі 3.11, однак виділення коридору відбувається на всьому ярусі. Тобто коли множини одного рангу сформовані, здійснюється виділення коридору як у множинах, так і між множинами одного рангу. Кількість оброблюваних векторів значно скоротилася (див. приклад 3.11).

Таким чином, процедуру A_0 зі стратегіями $\{L_w\}$ і правилами K_w можна покласти в основу побудови наближених і точних алгоритмів розв'язання задачі про рюкзак, тобто одержати різні модифікації процедури A_0 залежно від комбінацій використовуваних правил відсікання $\{L_w\}$ безперспективних шляхів у множинах μ_{sj}^r на основі застосування принципу оптимізації по напрямку (рис. 3.17).

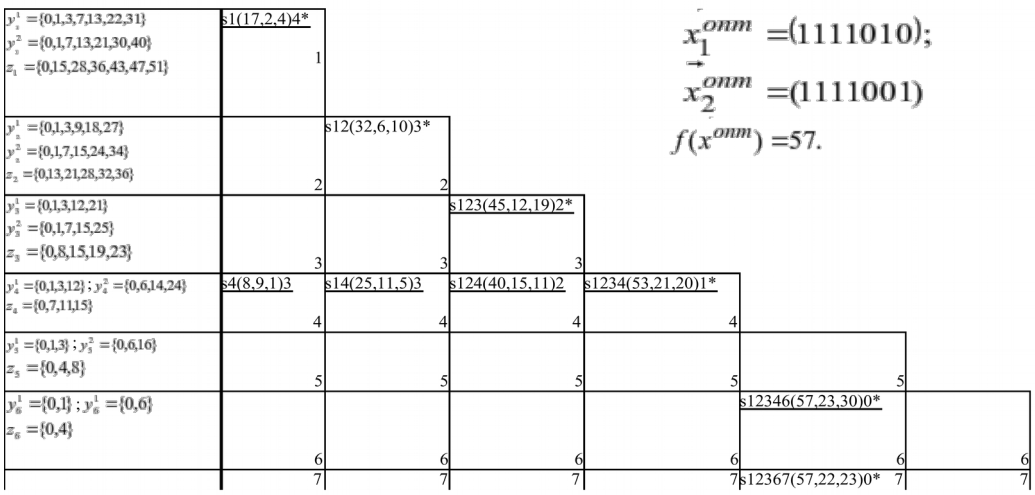


Рис. 3.17. Ілюстрація роботи стратегій L_9 і L_{13}

3.4. Алгоритми розв'язання задачі 0,1-рюкзак на основі рангового підходу

Алгоритми методу відсікань безперспективних варіантів

На основі висунутих у п. 3.3 стратегій $\{L_w\}$ і правил $\{K_w\}$ необхідно розробити алгоритми розв'язання задачі цілочисельного лінійного програмування 0,1-рюкзак. У результаті отримано одноетапні й багатоетапні алгоритми, класифікація яких подана на рис. 3.18.

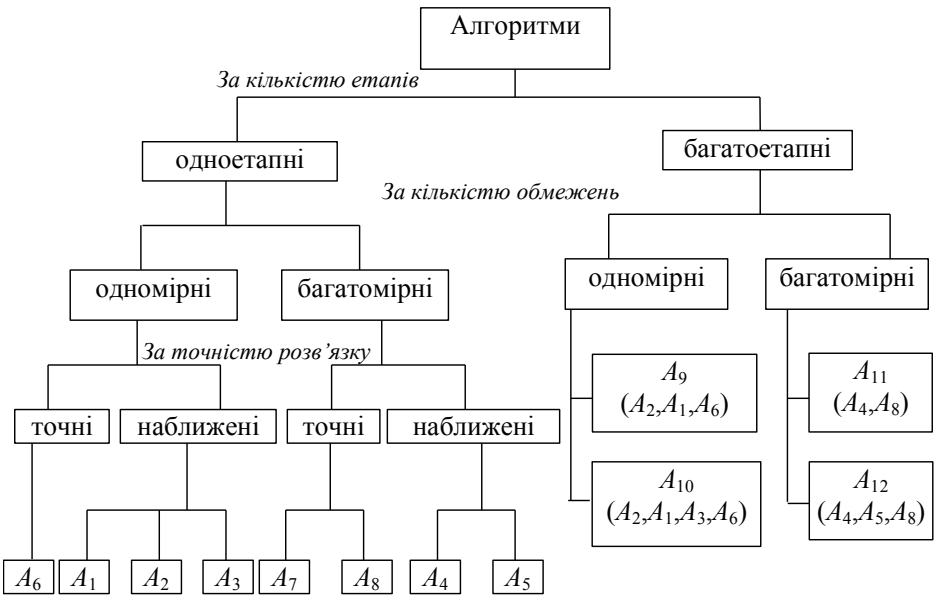


Рис. 3.18. Класифікація алгоритмів-модифікацій узагальненої процедури A_0

Основними кваліфікаційними ознаками є кількість етапів і точність етапів розв'язання задачі. За першою ознакою алгоритми поділяються на одноетапні й багатоетапні. Даний поділ означає, що вихідна задача може

розв'язуватися один (одноетапні) або кілька разів (багатоетапні) різними алгоритмами, перелік яких представлений у дужках. Друга ознака визначає похибку одержуваного припустимого розв'язку задачі (3.1) – (3.3) по вагах функціонала від оптимального розв'язку. Якщо алгоритм принципово не може дати точного розв'язку по вагах функціонала, то такий алгоритм будемо називати наближеним.

3.4.1. Наближені алгоритми розв'язання задачі ЦЛП із БЗ

Алгоритм А1

Розгляд наближених алгоритмів почнемо з одномірної задачі (3.4)–(3.6). Найбільш простим є алгоритм A_1, що побудований на основі процедури A_0 і реалізує стратегії відсікань L_1 і L_3. Його покроковий опис має такий вигляд.

Крок 1. З вершини *s* будуються множини шляхів $m_{sj}^{r=1}, j=(\overline{1,n})$ першого рангу r, що задовольняють властивість ν, і визначаються в множинах $m_{sj}^{r=1}$ шляхи максимальної довжини $\left|\mu_{sj}^{*r}\right|$ по вагах функціонала c_j. Для кожної вершини j визначається вага γ_j за правилом (3.19).

Крок 2. Виключаються шляхи $\left|\mu_{sp}^{r}\right|$, $p=(\overline{r,n})$ в множинах m_{sj}^{r} поточного рангу r, довжини яких $d_c\left(\mu_{sp}^{r}\right)$ задовольняють нерівність (3.20).

Крок 3. Формуються множини шляхів $m_{sp}^{r=r+1}$, $p=(\overline{1,n})$ наступного рангу, що задовольняють властивість ν, на базі множин шляхів m_{sj}^{r} попереднього рангу на основі рекурентного співвідношення (3.18). В утворених множинах $m_{sp}^{r=r+1}$ виділяємо найдовші шляхи $\left|\mu_{sp}^{*r=r+1}\right|$. Якщо виявиться кілька шляхів максимальної довжини, то серед них вибирається шлях з меншим значенням довжини по вагах обмежень a_{1j}.

Крок 4. Перевіряється, чи всі множини шляхів наступного (r+1)–го рангу порожні. Якщо це так, то переходимо до кроку 5, якщо ні, то перевіряємо $r = (n-1)$. У випадку виконання рівності переходимо до кроку 5, інакше збільшуємо r на 1 і виконуємо крок 2.

Крок 5. Виділяємо в множині $\left|\mu_{sj}^{*r=1}, \mu_{sp}^{*r=2}, \ldots, \mu_{sw}^{*r=n}\right|$ шлях максимальної довжини й алгоритм A_1 закінчує роботу.

Одержання наближеного розв'язку обумовлено застосуванням стратегії вибору L_1. Приклад 3.2, наведений у попередньому пункті ілюструє роботу алгоритму A_1.

Алгоритм А2

Іншою стратегією вибору шляхів є стратегія L_4, що за допомогою алгоритму A_2 дозволяє одержати шляхи максимально можливого рангу r у

графі $D\Delta$. Її можна реалізувати, якщо в $D\Delta$ визначати найкоротші шляхи по вагах обмежень на основі процедури A_0 між вершиною s і всіма іншими вершинами графа $D\Delta$, використовуючи при цьому рекурентне співвідношення (3.21), і, природно, відсікання за правилом L_3.

Покроковий опис алгоритму A_2 має такий вигляд.

Крок 1. З вершини s будуються множини шляхів $m_{sj}^{r=1}, j=\overline{(1,n)}$ першого рангу r, що задовольняють властивість ν, і визначаються в множинах $m_{sj}^{r=1}$ шляхи максимальної довжини $\left\{\mu_{sj}^{*r}\right\}$ по вагах функціонала c_j. Для кожної вершини j визначається вага γ_j за правилом (3.19).

Крок 2. Виключаються шляхи $\left\{\mu_{sp}^{r}\right\}$, $p=\overline{(r,n)}$ в множинах m_{sj}^{r} поточного рангу r, довжини яких $d_c\left(\mu_{sp}^{r}\right)$ задовольняють нерівність (3.20).

Крок 3. Формуються множини шляхів $m_{sp}^{r=r+1}$, $p=\overline{(1,n)}$ наступного рангу, що задовольняють властивість ν, на базі множин шляхів m_{sj}^{r} попереднього рангу, які мають мінімальне значення в m_{sj}^{r} довжини по вагах обмежень, на основі рекурентного співвідношення (3.21). В утворених множинах $m_{sp}^{r=r+1}$ виділяємо найдовші шляхи $\left\{\mu_{sp}^{*r=r+1}\right\}$.

Крок 4. Перевіряється, чи всі множини шляхів наступного (r+1)–го рангу порожні. Якщо це так, то переходимо до кроку 5, якщо ні, то перевіряємо r = (n–1). У випадку виконання рівності переходимо до кроку 5, інакше збільшуємо r на 1 і виконуємо крок 2.

Крок 5. Виділяємо в множині $\left\{\mu_{sj}^{*r=1},\mu_{sp}^{*r=2},\ldots,\mu_{sw}^{*r=n}\right\}$ шлях максимальної довжини й номер останньої вершини w, що належить цьому шляху, після чого алгоритм A_2 закінчує роботу.

Приклад 3.3 відображає хід розв'язання за алгоритмом A_2.

Алгоритм А3

Виходячи з поняття коридору й у відповідності зі стратегією L_7 пропонується такий наближений алгоритм A_3 розв'язання задачі (3.10) – (3.12), що полягає в тім, щоб із множини m_{sj}^{r} рангу r у множині $m_{sj}^{r=r+1}, j=\overline{(r+1,n)}$ наступного рангу вибиралося не більше двох шляхів, що задовольняють властивість ν.

Першим є шлях з максимальною вагою по функціоналу c_j (за рекурентним співвідношенням (3.18)), а другим – шлях з найменшою вагою по обмеженню $\{a_{1j}\}$ (за рекурентним співвідношенням (3.21)), що відповідним чином змінить крок 3 алгоритму A_1. В усьому іншому покроковий опис алгоритму A_3 повністю збігається з описом алгоритму A_1. Втрата оптимального розв'язку обумовлена його потраплянням усередину

коридору. Тому, як видно з наступного прикладу, даний алгоритм A_3 є наближеним.

Приклад 3.3. Потрібно максимізувати функціонал

$$f(x) = 20x_1 + 19x_2 + 17x_3 + 13x_4 + 12x_5 + 10x_6 + 7x_7 + 5x_8 + 2x_9$$

при обмеженні $9x_1 + 8x_2 + 8x_3 + 4x_4 + 3x_5 + 2x_6 + 1x_7 + 5x_8 + 1x_9 \le 30$ алгоритмом A_3 (рис. 3.5).

Розв'язання. Порядок побудови шляхів у графі $D\Delta$ описаний у попередньому розділі. З кожної множини m_{sj}^r вибираються тільки два шляхи: максимальний по вагах функціонала й мінімальний по вагах обмеження (ці шляхи підкреслені), а іншими – нехтують. Із цієї причини в множині $m_{s5}^{r=5}$ виключився з подальшого розгляду шлях $\mu_{s145}^{r=3}$, що належить оптимальному, тому що домінували за алгоритмом A_3 шляхи $\mu_{s125}^{r=3}$ й $\mu_{s245}^{r=3}$. Отже й результатом розв'язання вийшов шлях $\mu_{s245679}^{r=6}$, що відповідає вектору $x = \{010111101\}$, що й дало відносну похибку $\Delta f = (64–63)/64 = 1.5\%$.

Алгоритм А4

Для багатомірної задачі (3.1)–(3.3) за аналогією також можна побудувати ефективні наближені алгоритми. Таким є алгоритм A_4, заснований на стратегії L_{10}, що з множини поточного рангу будує в множини наступного рангу $r=r+1$ тільки шлях з максимальною довжиною по вагах функціонала $\{c_j\}$. При цьому часова складність алгоритму зростає в $2m$ раз, тому що необхідно виконати для кожного вектора m операцій додавання й m операцій порівняння, щоб переконатися: задовольняє вектор, що відповідає цьому шляху, обмеження (3.8) чи ні. Отже й покроковий опис для алгоритму A_4 збігається з алгоритмом A_1. Приклад 3.9 пояснює розв'язання задачі алгоритмом A_4 та стратегії L_3 і L_{10}.

Алгоритм A_5

Застосування стратегії L_{11} дозволило побудувати алгоритм A_5, суть якого полягає у виборі з кожної множини m_{sj}^r поточного рангу r одного шляху, максимального по вагах функціонала й, у найгіршому разі, m шляхів μ_{sj}^r, мінімальних по вагах обмежень, що задовольняють властивість ν. Тоді, на відміну від алгоритму A_1, покроковий опис алгоритму A_5 зміниться на рис. 3.19.

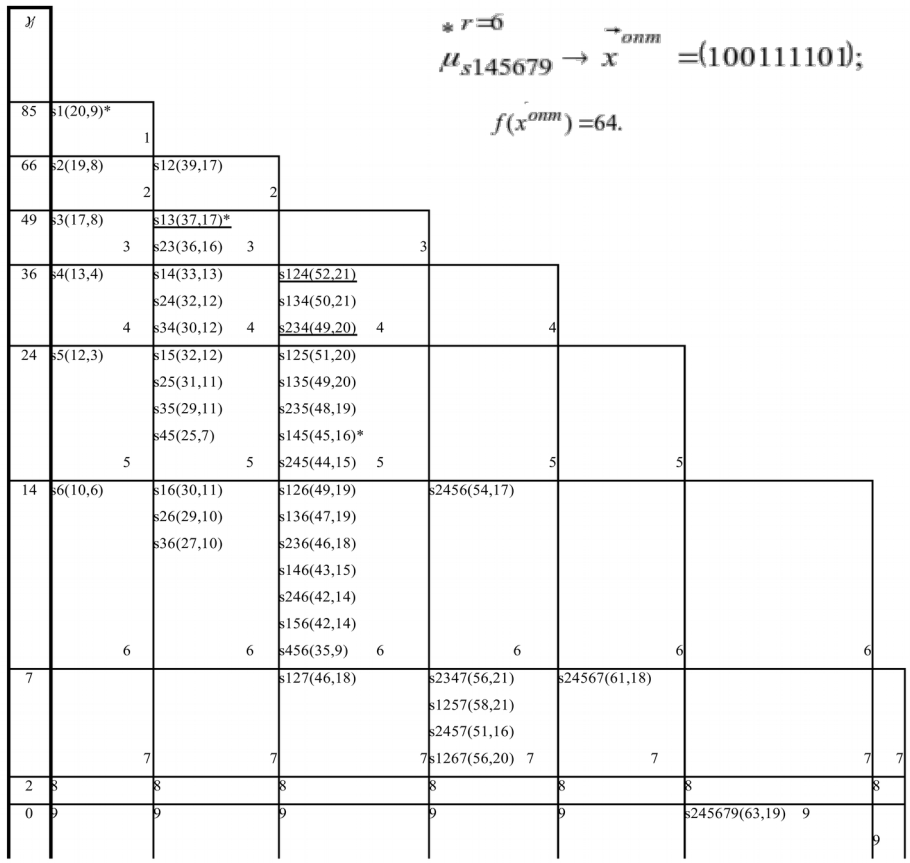


Рис. 3.19. Ілюстрація роботи алгоритму A_3

Крок 1. З вершини *s* будуються множини шляхів $m_{sj}^{r=1}, j=(\overline{1,n})$ першого рангу r, що задовольняють властивість ν, і визначаються в множинах $m_{sj}^{r=1}$ шляхи максимальної довжини $\left\{\mu_{sj}^{*r}\right\}$ по вагах функціонала c_j. Для кожної вершини j визначається вага γ_j за правилом (3.9).

Крок 2. Виключаються шляхи $\left\{\mu_{sp}^{r}\right\}$, $p=(\overline{r,n})$ в множинах m_{sj}^{r} поточного рангу r, довжини яких $d_c\left(\mu_{sp}^{r}\right)$ задовольняють нерівність (3.20).

Крок 3. Формуються множини шляхів $m_{sp}^{r=r+1}$, $p=(\overline{1,n})$ наступного рангу, що задовольняють властивість ν, на базі множин шляхів m_{sj}^{r} попереднього рангу на основі рекурентних співвідношень (3.18), (3.30). В утворених множинах $m_{sp}^{r=r+1}$ виділяємо найдовші шляхи $\left\{\mu_{sp}^{*r=r+1}\right\}$. Якщо виявиться кілька шляхів максимальної довжини, то серед них вибирається шлях з меншим значенням довжини по вагах обмеження $\{a_{ij}\}$. Якщо виявиться кілька шляхів з мінімальним значенням за *i*–м обмеженням, то вибирається той, у якого більше довжина по вагах функціонала.

Крок 4. Перевіряється, чи всі множини шляхів наступного (*r*+1)-го рангу порожні. Якщо це так, то переходимо до кроку 5, якщо ні, то

перевіряємо $r=(n-1)$. У випадку виконання рівності переходимо до кроку 5, інакше збільшуємо r на 1 і виконуємо крок 2.

Крок 5. Виділяємо в множині $\left\{\mu_{sj}^{*r=1}, \mu_{sp}^{*r=2}, ..., \mu_{sw}^{*r=n}\right\}$ шлях максимальної довжини й алгоритм A_5 закінчує роботу.

Цей алгоритм має одну невизначеність у тому випадку, коли в множині виявиться кілька векторів з максимальною довжиною по вагах функціонала й з різними значеннями по вагах обмежень, тобто, наприклад, по вагах першого обмеження більше другий вектор, по вагах другого – перший, по вагах третього – знову другий й т. д. Тому для запобігання цієї невизначеності при рівності довжин по вагах функціонала вибирається шлях з меншим значенням по вагах усіх обмежень. Якщо таке неможливо, то вибирається кожний. Може виявитися, що відкинутий шлях буде належати оптимальному шляху. Ця обставина і є однією з причин одержання алгоритмом A_5 наближеного розв'язку.

Приклад 3.10 ілюструє розв'язання задачі алгоритмом A_5.

3.4.2. Точні алгоритми розв'язання задачі ЦЛП із БЗ

Побудова точних алгоритмів ґрунтується на понятті коридору й використанні стратегій L_6, L_8, L_9, L_{12}, L_{13} відсікання. Для точних алгоритмів аналітична оцінка часової складності й величини апаратурних витрат представляє труднощі через NP–повноту [11] розв'язуваної задачі. Основною причиною неможливості оцінки є випадкова кількість векторів усередині виділюваного коридору. Природа випадковості кількості векторів усередині коридору визначається співвідношеннями між коефіцієнтами функціонала c_j і коефіцієнтами обмежень a_{ij}, а також приростами цих співвідношень між j-ми компонентами двійкового вектора $x = x_1,...,x_n$ при заданому законі розподілу коефіцієнтів у функціоналі й обмеженнях.

Алгоритм A_6

На основі стратегій L_6, L_8, L_9 побудуємо точний алгоритм A_6 розв'язання одномірної задачі (3.10) – (3.12), покроковий опис якого буде мати такий вигляд.

Крок 1. З вершини s будуються множини шляхів $m_{sj}^{r=1}, j=\overline{(1,n)}$ першого рангу r, що задовольняють властивість ν, і визначаються в множинах $m_{sj}^{r=1}$ шляхи максимальної довжини $\left\{\mu_{sj}^{*r}\right\}$ по вагах функціонала c_j. Для кожної вершини j формуються калібровані вектори y_j (3.24) і z_j (3.27).

Крок 2. Виключаються шляхи в множинах m_{sj}^{r} поточного рангу r, довжини яких $d_c\left(\mu_{sp}^{r}\right)$ задовольняють нерівність (3.20).

Крок 3. Для кожного шляху $\left|\mu_{sj}^{r}\right|$, $j=(\overline{r,n})$ поточного рангу r визначається за правилом K_1 значення $\hat{r}_в$. Виключаються шляхи $\mu_{sj}^{r}\in m_{sj}^{r}$, довжини яких задовольняють нерівність (3.29).

Крок 4. Формуються множини шляхів $m_{sp}^{r=r+1}$, $p=(\overline{1,n})$ наступного рангу, що задовольняють властивість ν, на базі множин шляхів m_{sj}^{r} попереднього рангу на основі принципу оптимізації по напрямку (3.17) з виділенням коридору за стратегією L_6 і виключенням векторів усередині коридору у відповідності зі стратегією L_8. Шлях у множині $m_{sp}^{r=r+1}$ може бути сформований, якщо він задовольняє властивість. Якщо властивість ν не виконується, то шлях виключається з подальшого аналізу. В утворених множинах $m_{sp}^{r=r+1}$ виділяємо найдовші шляхи $\left|\mu_{sp}^{*r=r+1}\right|$.

Крок 5. Перевіряється, чи всі множини шляхів наступного (r+1)-го рангу порожні. Якщо це так, то переходимо до кроку 6, якщо ні, то перевіряємо $r = (n–1)$. У випадку виконання рівності переходимо до кроку 6, інакше збільшуємо r на 1 і виконуємо крок 2.

Крок 6. Виділяємо в множині $\left|\mu_{sj}^{*r=1},\mu_{sp}^{*r=2},\ldots,\mu_{sw}^{*r=n}\right|$ шлях максимальної довжини й алгоритм A_6 закінчує роботу.

Приклад 3.8 пояснює розв'язання задачі (3.10)–(3.12) алгоритмом A_6. Як видно з цього прикладу, у кожній множині m_{sj}^{r} залишаються тільки шляхи, відсортовані й у порядку зменшення по вагах функціонала й у порядку зменшення по вагах обмеження (вони підкреслені). Інші виключаються з подальшого аналізу.

Алгоритм A_7

Для розв'язання багатомірної задачі ЦЛП із БЗ (3.7)–(3.9) розглянемо два точних алгоритми A_7 і A_8, засновані на понятті m–мірного коридору й стратегій вибору L_{12}, L_{13} відповідно. Принципове розходження алгоритмів A_7 і A_8 полягає в тому, що перший виділяє m-мірний коридор усередині множини поточного рангу r графа $D\Delta$, а другий виділяє m-мірний коридор безпосередньо на всьому ярусі графа DΔ. Відсікання, засновані на стратегії L_9 і правилах K_2, K_3, K_4, залишаються однаковими для обох алгоритмів. Покроковий опис алгоритму A_7 має такий вигляд.

Крок 1. З вершини *s* будуються множини шляхів $m_{sj}^{r=1}$, $j=(\overline{1,n})$ першого рангу r, що задовольняють властивість ν, і визначаються в множинах $m_{sj}^{r=1}$ шляхи максимальної довжини $\left|\mu_{sj}^{*r}\right|$ по вагах функціонала c_j. Для кожної вершини j формуються калібровані вектори y_j^i (3.31) і z_j (3.27).

Крок 2. Для кожного шляху $\left|\mu_{sj}^{r}\right|$, $j=(\overline{r,n})$ поточного рангу r визначається за правилом K_1 значення $\hat{r}_в$. Виключаються шляхи $\mu_{sj}^{r}\in m_{sj}^{r}$, довжини яких задовольняють нерівність (3.29).

Крок 3. Формуються множини шляхів $m_{sp}^{r=r+1}$, $p=(\overline{1,n})$ наступного рангу, що задовольняють властивість ν, на базі множин шляхів m_{sj}^{r} попереднього рангу на основі принципу оптимізації по напрямку (3.17) з виділенням m-мірного коридору у вигляді підмножини $m_{sp}^{rk}\in m_{sp}^{r=r+1}$ у відповідності зі стратегією L_{12} і відсіканням безперспективних векторів за правилом K_4. Шлях у множині $m_{sp}^{r=r+1}$ може бути сформований, якщо він задовольняє властивість. Якщо властивість ν не виконується, то шлях виключається з подальшого аналізу. В утворених множинах m_{sp}^{rk} виділяємо найдовші шляхи $\left|\mu_{sp}^{*r=r+1}\right|$.

Крок 4. Перевіряється, чи всі множини шляхів наступного (r+1)-го рангу порожні. Якщо це так, то переходимо до кроку 5, якщо ні, то перевіряємо r=(n–1). У випадку виконання рівності переходимо до кроку 5, інакше збільшуємо r на 1 і виконуємо крок 2.

Крок 5. Виділяємо в множині $\left|\mu_{sj}^{*r=1},\mu_{sp}^{*r=2},\ldots,\mu_{sw}^{*r=n}\right|$ шлях максимальної довжини й алгоритм A_7 закінчує роботу.

Приклад 3.11 відображає розв'язання задачі алгоритмом A_7. Головною перевагою цього алгоритму є одержання точного розв'язку. Однак це досягається за рахунок обробки великої кількості векторів, що робить його малопридатним для практичного використання.

Алгоритм A_8

З метою усунення цього недоліку пропонується алгоритм A_8, що здійснює виділення m–мірного коридору на всьому ярусі. Суть цього алгоритму полягає в тому, що якщо на ярусі в множинах, що лежать у графі $D\Delta$ вище деякого шляху μ_{sp}^{*r} рангу r, існує шлях μ_{sp}^{**r}, що має не менше значення по вагах функціонала й менше значення по вагах усіх обмежень, то такий шлях $m_{sp}^{r=r+1}$ не потрапляє в m-мірний коридор на ярусі. У цьому випадку покроковий опис A_8 набуде такого вигляду.

Крок 1. З вершини s будуються множини шляхів $m_{sj}^{r=1}$, $j=(\overline{1,n})$ першого рангу r, що задовольняють властивість ν, і визначаються в множинах $m_{sj}^{r=1}$ шляхи максимальної довжини $\left|\mu_{sj}^{*r}\right|$ по вагах функціонала c_j. Для кожної вершини j формуються калібровані вектори y_j (3.31) і z_j (3.27).

Крок 2. Виключаються шляхи в множинах m_{sj}^{r} поточного рангу r, довжини яких $d_c\left(\mu_{sp}^{r}\right)$ задовольняють нерівність (3.20).

Крок 3. Формуються множини шляхів $m_{sp}^{r=r+1}$, $\mathrm{p}=\overline{(1,\mathrm{n})}$ наступного рангу, що задовольняють властивість ν, на базі множин шляхів m_{sj}^{r} попереднього рангу на основі принципу оптимізації по напрямку (3.17). Шлях у множині $m_{sp}^{r=r+1}$ може бути сформований, якщо він задовольняє властивість. Якщо властивість ν не виконується, то шлях виключається з подальшого аналізу. В утворених множинах $\mathrm{m}_{\mathrm{sp}}^{\mathrm{r}=r+1}$ виділяємо локальні екстремуми $\left\{\mu_{sp}^{*r=r+1}\right\}$.

Крок 4. Виконуємо процедуру виділення m–мірного коридору на всьому сформованому ранзі r=r+1. Для цього послідовно перевіряються всі множини $m_{sp}^{r=r+1}$, $\mathrm{p}=\overline{(\mathrm{r}+1,\mathrm{n})}$ у відповідності зі стратегією L_{13} і виробляється відсікання безперспективних векторів за правилом K_4.

Крок 5. Перевіряється, чи всі множини шляхів наступного (r+1)–го рангу порожні. Якщо це так, то переходимо до кроку 6, якщо ні, то перевіряємо r=(n–1). У випадку виконання рівності також переходимо до кроку 6, інакше збільшуємо r на 1 і виконуємо крок 2.

Крок 6. Виділяємо в множині $\left\{\mu_{sj}^{*r=1},\mu_{sp}^{*r=2},\ldots,\mu_{sw}^{*r=n}\right\}$ шлях максимальної довжини й алгоритм A_8 закінчує роботу.

У прикладах 3.11 і 3.12 та сама задача розв’язувалася алгоритмами A_7 і A_8 відповідно. Однак кількість векторів, оброблюваних алгоритмом A_8, у два рази менше, ніж алгоритмом A_7.

Таким чином, на основі принципу оптимізації по напрямку в дискретному просторі станів, обумовленим графом D (рис. 3.20), удалося побудувати точні алгоритми для розв’язання як одномірної, так і багатомірної задачі ЦЛП із БЗ. Покажемо, що для зниження загальної часової складності, за рахунок звуження ширини коридору по вагах функціонала, доцільно використовувати багатоетапну фільтрацію.

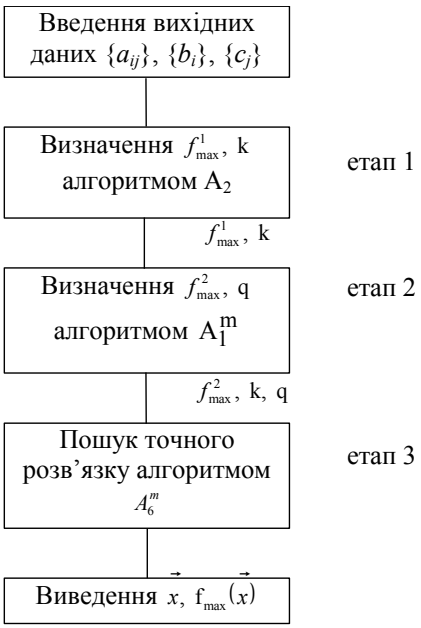


Рис. 3.20. Структура алгоритму А9

3.4.3. Багатоетапні алгоритми розв'язання задачі ЦЛП із БЗ

Багатоетапні алгоритми призначені для зниження загальної часової складності за рахунок звуження ширини коридору по вагах функціонала. Основою побудови багатоетапних алгоритмів служить багатоетапна фільтрація, у якій можна виділити два основних етапи. Перший етап – це визначення припустимого наближеного розв'язку f^*, що використовується на другому етапі для пошуку оптимального розв'язку точним алгоритмом, що призведе до підвищення ефективності фільтрації безперспективних шляхів у коридорі.

Алгоритм A_9

Спочатку побудуємо багатоетапні алгоритми для одномірної задачі (3.10)–(3.12). На рис. 3.20 показаний трьохетапний алгоритм A_9, у якому на першому етапі задача розв'язується алгоритмом A_2 з використанням додатково стратегії L_3. Застосування цього алгоритму дозволить швидко знайти припустимий розв'язок x_1 й запам'ятати $f^1_{\max} = f(x_1)$, а також визначити у відповідності зі стратегією L_5 величину k (3.22), що дозволить на наступних етапах не будувати шляхи першого рангу в множинах $m^{r=1}_{sj}, j = (\overline{k+1, n})$. На другому етапі задача розв'язується алгоритмом A_1 з урахуванням величини функціонала отриманого припустимого розв'язку $f_{\max}$ і значення k. Це забезпечить, як видно з прикладу 3.13, більш ефективне відсікання за стратегію L_3, і виключать на першому ранзі безперспективні множини за стратегією L_5. Необхідність застосування зазначених стратегій

відсікання призведе до зміни 1, 2 і 5 кроків алгоритму A_1, у результаті чого утвориться алгоритм A_1^m, покроковий опис якого набуває такого вигляду.

Алгоритм A_1^m

Крок 1. З вершини *s* будуються множини шляхів $m_{sj}^{r=1}, j=\overline{(1,n)}$ першого рангу *r*, що задовольняють властивість ν, і визначаються в множинах $m_{sj}^{r=1}$ локальні екстремуми $\left|\mu_{sj}^{*r}\right|$.

Крок 2. Виключаються шляхи $\left|\mu_{sp}^{r}\right|, p=\overline{(r,n)}$ в множинах $\mathrm{m}_{\mathrm{sj}}^{\mathrm{r}}$ поточного рангу r, довжини яких $d_c(\mu_{sp}^r)$ задовольняють нерівність

$$d_c\left(\mu_{sp}^r\right)+\gamma_p<\max\left\{c_j\right\}\left|d_c\left(\mu_{sp}^{*r}\right), f_{\max}^1\right| \qquad (3.33)$$

Крок 3. Формуються множини шляхів $m_{sp}^{r=r+1}, p=\overline{(1,n)}$ наступного рангу, що задовольняють властивість ν, на базі множин шляхів m_{sj}^r попереднього рангу на основі рекурентного співвідношення (3.18). В утворених множинах $m_{sp}^{r=r+1}$ виділяємо найдовші шляхи $\left|\mu_{sp}^{*r=r+1}\right|$. Якщо виявиться кілька шляхів максимальної довжини, то серед них вибирається шлях з меншим значенням довжини по вагах обмежень a_{1j}.

Крок 4. Перевіряється, чи всі множини шляхів наступного (*r*+1)-го рангу порожні. Якщо це так, то переходимо до кроку 5, якщо ні, то перевіряємо *r*=(*n*–1). У випадку виконання рівності переходимо до кроку 5, інакше збільшуємо r на 1 і виконуємо крок 2.

Крок 5. Виділяємо в множині $\left|\mu_{sj}^{*r=1}, \mu_{sp}^{*r=2}, \ldots, \mu_{sw}^{*r=n}, f(\vec{x}_1)\right|$ шлях максимальної довжини й запам'ятовуємо ранг цього шляху q. Після цього алгоритм A_1^m закінчує роботу.

Ефективність використання даного етапу залежить від точності припустимого розв'язку x_2. Він також дозволяє визначити величину q, що служить для відсікання безперспективних розв'язків на третьому етапі у відповідності зі стратегією L_2.

На третьому етапі задача розв'зується точним алгоритмом A_6 з урахуванням стратегій відсікань L_2 і L_5. Застосування цього етапу відповідно до твердження 3.3 і теореми 3.4 гарантує одержання точного розв'язку. Покроковий опис алгоритму A_6 зміниться, що призведе до одержання його модифікації – алгоритму A_6^m, що має такий вигляд.

Крок 1. З вершини *s* будуються множини шляхів $m_{sj}^{r=1}, j=\overline{(1,k)}$ першого рангу r, що задовольняють властивість ν, і визначаються в

множинах $m_{sj}^{r=1}$ шляхи максимальної довжини $\left|\mu_{sj}^{*r}\right|$ по вагах функціонала c_j.

Крок 2. Для кожного шляху $\left|\mu_{sj}^{r}\right|$, $j=(\overline{r,n})$ поточного рангу r визначається за правилом K_1 значення $\hat{r}_в$. Виключаються шляхи в множинах m_{sj}^{r} поточного рангу, довжини яких задовольняють нерівність

$$d_c\left(\mu_{sp}^{r}\right)+\gamma_p<\max\left|d_{c\left|c_j\right|}\left(\mu_{sp}^{*r}\right),\ f_{\max}^{1},\ f_{\max}^{2}\right|. \qquad (3.34)$$

Крок 3. У відповідності зі стратегією L_2 для кожного шляху μ_{sj}^{r} поточного рангу r виробляється перевірка виконання умови

$$r+\hat{r}_в<q. \qquad (3.35)$$

Якщо умова виконуються, то шлях $\mu_{sj}^{r}\in m_{sj}^{r}$ виключається з подальшого аналізу.

Крок 4. Формуються множини шляхів $m_{sp}^{r=r+1}$, $p=(\overline{r+1,n})$ наступного рангу, що задовольняють властивість ν, на базі множин шляхів m_{sj}^{r} попереднього рангу на основі принципу оптимізації по напрямку (3.11) з виділенням коридору за стратегією L_6 і виключенням векторів усередині коридору у відповідності зі стратегією L_8. Шлях у множині $m_{sp}^{r=r+1}$ може бути сформований, якщо він задовольняє властивість. Якщо властивість ν не виконується, то шлях виключається з подальшого аналізу. В утворених множинах $m_{sp}^{r=r+1}$ виділяємо найдовші шляхи $\left|\mu_{sp}^{*r=r+1}\right|$.

Крок 5. Перевіряється, чи всі множини шляхів наступного (r+1)–го рангу порожні. Якщо це так, то переходимо до кроку 6, якщо ні, то перевіряємо $r=(n-1)$. У випадку виконання рівності переходимо до кроку 6 інакше збільшуємо r на 1 і виконуємо крок 2.

Крок 6. Виділяємо в множині $\left|\mu_{sj}^{*r=1},\mu_{sp}^{*r=2},\ldots,\mu_{sw}^{*r=n},f(\vec{x}_1),f(\vec{x}_2)\right|$ шлях максимальної довжини й алгоритм A_6^m закінчує роботу.

Алгоритм A_{10}

Неважко побачити, що чим більш точний розв'язок отримується при розв'язанні задачі ЦЛП із БЗ наближеними алгоритмами, тим менше припустимих розв'язків доводиться аналізувати на останньому етапі й тим швидше буде знайдено точний розв'язок і доведено його оптимальність. Найточнішим з наближених алгоритмів є алгоритм A_3.

Тому пропонується чотирьохетапний алгоритм A_{10}, структурна схема якого зображена на рис. 3.21.

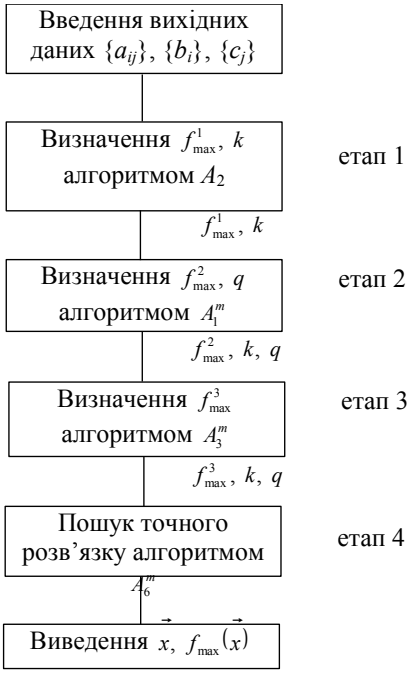


Рис. 3.21. Структура алгоритму A_{10}

Перші два етапи ідентичні першим двом етапам алгоритму A_9. На третьому етапі використовується алгоритм A_3, що реалізує стратегію L_7. Для нього також справедливі стратегії відсікань L_2, L_5, що призведе до створення алгоритму A^m_3. Для нього перші три кроки збігаються з першими трьома m кроками алгоритму A^m_6, а другі три відповідають крокам 3, 4 і 5 алгоритму A_3 з еквівалентною заміною адрес переходів у ньому.

Результатом розв'язання задачі алгоритмом A^m_3 є вектор x_3 і відповідне йому значення функціонала (3.10), що дорівнює $f_{\max} = f(x_3)$. На заключному, четвертому, етапі задача розв'язується алгоритмом A^m_6. Однак на кроці 6 при виборі оптимального розв'язку буде враховуватися вже n+3 припустимих розв'язків.

Приклад чотирьохетапного алгоритму показаний на прикладі 3.2.

Алгоритм A_{11}

Розглянемо можливість побудови багатоетапних алгоритмів для багатомірної задачі ЦЛП із БЗ (3.7)–(3.9). У роботі пропонується два таких алгоритми – A_{11} і A_{12}. Перший алгоритм A_{11}, як показано на рис. 3.22, є двохетапним.

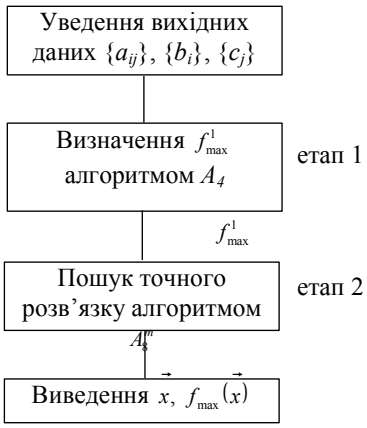


Рис. 3.22. Структура алгоритму А11

На першому етапі швидкого пошуку припустимого розв'язку використовується алгоритм A_4, заснований на стратегіях L_1, L_9, L_{10}. Покроковий опис алгоритму A_4^m має такий вигляд.

Крок 1. З вершини *s* будуються множини шляхів $m_{sj}^{r=1}, j=(\overline{1,n})$ першого рангу *r*, що задовольняють властивість ν, і визначаються в множинах $m_{sj}^{r=1}$ шляхи максимальної довжини $\left|m_{sj}^{*r}\right|$ по вагах функціонала c_j. Для кожної вершини j формуються калібровані вектори y_j^i (3.31) і z_j (3.27).

Крок 2. Для кожного шляху $\left|\mu_{sj}^{r}\right|$, $j=(\overline{r,n})$ поточного рангу r визначається за правилом K_3 і виразом (3.26) значення $\hat{r}_в$. Виключаються шляхи в множинах m_{sj}^{r} поточного рангу *r*, довжини яких $d_c(\mu_{sp}^{r})$ задовольняють нерівність (3.29).

Крок 3. Формуються множини шляхів $m_{sp}^{r=r+1}$, $p=(\overline{r+1,n})$ наступного рангу, що задовольняють властивість ν, на базі множин шляхів m_{sj}^{r} попереднього рангу на основі рекурентного співвідношення (3.18). В утворених множинах $m_{sp}^{r=r+1}$ виділяємо найдовші шляхи $\left|\mu_{sp}^{*r=r+1}\right|$. Якщо виявиться кілька шляхів максимальної довжини, то серед них вибирається шлях з меншим значенням довжини по вагах обмежень a_{1j}.

Крок 4. Перевіряється, чи всі множини шляхів наступного (*r*+1)–го рангу порожні. Якщо це так, то переходимо до кроку 5, якщо ні, то перевіряємо $r=(n-1)$. У випадку виконання рівності переходимо до кроку 5, інакше збільшуємо r на 1 і виконуємо крок 2.

Крок 5. Виділяємо в множині $\left\{\mu_{sj}^{*r=1}, \mu_{sp}^{*r=2}, \ldots, \mu_{sw}^{*r=n}\right\}$ шлях максимальної довжини й алгоритм A_4^m закінчує роботу.

Алгоритмом A_4^m буде знайдено припустимий розв'язок задачі x_1 й значення функціонала (3.7), що дорівнює $f_{\max}^1 = f(\bar{x}_1)$, що як параметр передається на другий етап алгоритму A_{11}.

Слід зазначити, що калібровані вектори, побудовані алгоритмом A_4^m, не змінюються і використовуються на другому етапі алгоритму A_8. Його покроковий опис зміниться через необхідність ураховувати $f_{\max}^1$, що призведе до модифікації – алгоритму A_8^m.

Крок 1. З вершини *s* будуються множини шляхів $m_{sj}^{r=1}, j = (\overline{1,n})$ першого рангу r, що задовольняють властивість ν, і визначаються в множинах $m_{sj}^{r=1}$ шляхи максимальної довжини $\left\{\mu_{sj}^{*r}\right\}$ по вагах функціонала c_j.

Крок 2. Для кожного шляху $\left\{\mu_{sj}^{r}\right\}$, $j = (\overline{r,n})$ поточного рангу r визначається за правилом K_3 і виразом (3.32) значення $\hat{r}_в$. Виключаються шляхи $\mu_{sj}^r \in m_{sj}^r$, довжини яких задовольняють нерівність

$$d_c\left(\mu_{sp}^r\right) + \hat{z}_{p_{\hat{r}_в}}^{в}\left(\mu_{sp}^r\right) < \max\left\{d_{cc_j}\left(\mu_{sp}^{*r}\right), f_{\max}^1\right\}. \qquad (3.36)$$

Крок 3. Формуються множини шляхів $m_{sp}^{r=r+1}, p = (\overline{r+1,n})$ наступного рангу, що задовольняють властивість ν, на базі множин шляхів m_{sj}^r попереднього рангу на основі принципу оптимізації по напрямку (3.17) з виділенням *m*-мірного коридору у вигляді підмножини $m_{sp}^{rk} \in m_{sp}^{r=r+1}$ у відповідності зі стратегією L_{12} і відсіканням безперспективних векторів за правилом K_4. Шлях у множині $m_{sp}^{r=r+1}$ може бути сформований, якщо він задовольняє властивість. Якщо властивість ν не виконується, то шлях виключається з подальшого аналізу. В утворених множинах m_{sp}^{rk} виділяємо найдовші шляхи $\left\{\mu_{sp}^{*r=r+1}\right\}$.

Крок 4. Перевіряється, чи всі множини шляхів наступного (*r*+1)-го рангу порожні. Якщо це так, то переходимо до кроку 5, якщо ні, то перевіряємо *r*=(*n*–1). У випадку виконання рівності переходимо до кроку 5, інакше збільшуємо r на 1 і виконуємо крок 2.

Крок 5. Виділяємо в множині $\left\{\mu_{sj}^{*r=1}, \mu_{sp}^{*r=2}, \ldots, \mu_{sw}^{*r=n}\right\}$ шлях максимальної довжини й алгоритм A_8^m закінчує роботу.

Алгоритм A_{12}

Щоб зменшити кількість оброблюваних векторів усередині m-мірного коридору, пропонується трьохетапний алгоритм A_{12} (рис. 3.23).

Перший етап відповідає виконанню алгоритму A_4^m, у результаті чого обчислюється припустимий розв'язок x_1 й $f_{\max}^1 = f(x_1)$. На другому етапі використовується алгоритм A_5^m, заснований на стратегіях L_9, L_{11}. Покроковий опис алгоритму A_5^m збігається з описом алгоритму A_8^m, за винятком кроку 3, що має такий вигляд.

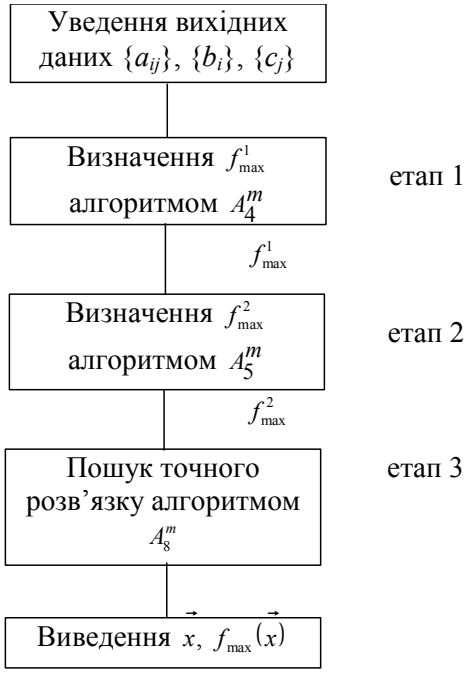


Рис. 3.23. Структура алгоритму А12

Крок 3. Формуються множини шляхів $m_{sp}^{r=r+1}$, $p = (\overline{r+1, n})$ наступного рангу, що задовольняють властивість ν, на базі множин шляхів m_{sj}^r попереднього рангу на основі принципу оптимізації по напрямку (3.17) і (3.21). В утворених множинах m_{sp}^{rk} виділяємо найдовші шляхи $\left\{\mu_{sp}^{*r=r+1}\right\}$. Якщо виявиться кілька шляхів максимальної довжини, то серед них вибирається шлях з найменшим значенням довжини по вагах обмежень (3.8). Якщо виявиться кілька шляхів з мінімальним значенням за *i*–м обмеженням, то вибирається той, у якого більше довжина по вагах функціонала.

На третьому етапі, як показано на рис. 3.23, виконується точний алгоритм A_8^m, у якого перевірка нерівності (3.36) на кроці 2 замінена перевіркою нерівності

$$d_c\left(\mu_{sp}^r\right) + \hat{z}^{в}_{p_{\hat{r}_в}}\left(\mu_{sp}^r\right) < \max\left\{d_c\left(c_j\,\mu_{sp}^{*r}\right),\ f_{\max}^1,\ f_{\max}^2\right\}. \qquad (3.37)$$

Приклади 3.3 і 3.4 демонструють порядок розв'язання алгоритмами A_{11} і A_{12} відповідно. Таким чином, багатоетапне розв'язання задачі забезпечує одержання швидкого припустимого розв'язку й гарантує точний результат за рахунок використання принципу оптимізації по напрямку й виділення m-мірного коридору.

3.4.4. Особливості розв'язання ЦЛП із БЗ у випадку рівності коефіцієнтів у функціоналі

Розглянемо випадок задачі (3.7) – (3.9), коли C_j = const для усіх $j = \overline{(1,n)}$, при цьому задача максимізації функціонала (3.1) при обмеженнях (3.8) – (3.9) зводиться до визначення шляху максимально можливого рангу в ациклічному графі D_Δ, оскільки максимальному рангу шляху в цьому випадку відповідає найбільше значення функціонала. Зрозуміло, що для побудови шляху максимально можливого рангу, довжина якого не перевищує деяку величину b_i, необхідно, щоб вона з додаванням кожного певного ребра збільшувалася на мінімально можливу величину. Тоді варто очікувати, що в шлях вдається включити найбільшу кількість ребер. Отже, що цікавить нас, шлях варто шукати серед множини найкоротших шляхів від вершини S до усіх інших вершин графа G з урахуванням обмежень (3.8) – (3.9). Позначимо через $\{\mu_s^{*r}(i)\}$ множину найкоротших шляхів при і–різних варіантах розподілу ваг у графі D_Δ. Тут $\mu_s^{*r}(i = \nu)$ $i = \overline{(1,m)}$ – найкоротший шлях максимально можливого рангу r для ν–варіанта розподілу ваг у графі, причому такої, що довжини цього шляху за всіма іншими варіантами розподілу ваг задовольняють обмеження (3.2), тоді задачу визначення шляху максимально можливого рангу, задовольняючи обмеження (3.8), можна визначити як відшукання $\max_r\{\mathrm{s\,r(i)}\}$ $\{\mu_s^{*r}(i)\}$ $i = \overline{(1,m)}$. З останнього випливає, що необхідно m раз шукати найкоротші шляхи, що задовольняють вираз (3.8) між вершиною S і іншими вершинами графа D_Δ і виділяти серед них шлях максимально можливого рангу. Потім в отриманій підмножині знайти шлях з максимальним значенням r. Однак у ряді випадків кількість побудов таких шляхів може бути різко скорочена, якщо виконуються умови, обумовлені наступним твердженням.

Твердження 3.5. Максимальний ранг шляху з вершини s дорівнює рангу найкоротшого шляху по кожному з і-варіантів розподілу ваг у графі D_Δ $i=\overline{(1,m)}$, якщо в і-варіанті його довжина досягла свого граничного значення b_i.

Доведення. Нехай у графі D_Δ при визначенні найкоротших шляхів між вершиною S і всіма іншими вершинам графа D_Δ для v-варіанта розподілу ваг виявилося, що шлях $\mu_s^{*r}(v)$ максимально можливого рангу r_v^* і його довжина досягли свого гранично можливого значення b_v, а довжини цього шляху, що залишилися (m–l)-варіантам розподілу ваг, задовольняють обмеження (3.2). Припустимо, що при k-варіанті розподілу ваг удалося побудувати найкоротший шлях $\mu_s^{*r}(k)$ рангу r_k^*, що задовольняє за обмеженнями (3.2) всі варіанти, що залишилися, розподілів ваг. Але тоді повинен існувати шлях $\mu_s^{*r}(v)$ рангу $r_v^*<r_k^*$, довжина якого $l\le b_v$, що суперечить первісному припущенню про те, що найкоротший шлях $\mu_s^{*r}(v)$ рангу r_v^* досягнув свого гранично можливого значення b_v. Отже, припущення про можливість існування шляху $\mu_s^{*r}(k)$ рангу $r_k^* > r_v^*$ при задоволенні обмежень (3.2) не є правильним, що й було потрібно довести. У такий спосіб виконання умов, що відповідають твердженням, дозволяє відразу відповістити на запитання розв'язуваної задачі.

Алгоритм розв'язання задачі

При визначенні шляху $\mu_s^{*r}(v)$ для v-варіанта розподілу ваг попередньо будемо здійснювати сортування вагових коефіцієнтів в v-обмеженні в порядку їхнього зростання й при цьому ребру, що входить у вершину 1 графа D_Δ, буде завжди відповідати найменший ваговий коефіцієнт, у вершину r – шлях більший і т. д., у вершину n – найбільший ваговий коефіцієнт. В основу розв'язання задачі буде покладена така спеціалізована процедура визначення найкоротших шляхів, складена з урахуванням того, що ваги в графі D_Δ закріплені за ребрами відповідно до зазначеного сортування.

Процедура $A^{'}$ визначення найкоротших шляхів у графі

Крок 1.1: v, на базі шляху S_1 формуємо всі можливі шляхи рангу r=2.

Крок 2. Для сформованих шляхів перевіряємо по усіх варіантах розподілу ваг $i=\overline{(1,m)}$ виконання нерівностей (3.2), якщо є серед них такі, для яких хоча б одна нерівність (3.2) не виконується, то ці шляхи виключаються з розгляду, а в тих, що залишилися, обчислюються довжини по усіх варіантах розподілу ваг і уточнюється, серед тих, що залишилися, чи є довжини шляхів гранично можливого значення, що досягнули, $i=\overline{(1,m)}$ b_i. Якщо так, то переходимо до кроку 5, а якщо ні, то виконуємо наступний крок.

Крок 3. Перевіряємо сформовану підмножину шляхів: порожня чи ні. Якщо підмножина порожня, то переходимо до кроку 6, якщо ні, то у

сформованій множині шляхів вибираємо найкоротший шлях за поточним варіантом розподілу ваг і переходимо до виконання наступного кроку.

Крок 4. На базі отриманого найкоротшого шляху формуємо всі можливі шляхи рангу $r=r+1$ і переходимо до кроку 2.

Крок 5. Закінчення роботи алгоритму й фіксація шляху, що досягнув свого граничного значення b_i.

Крок 6. Закінчення роботи алгоритму й фіксація найкоротшого шляху, отриманого останнім разом на кроці 4.

З урахуванням розробленої процедури A' визначення найкоротших шляхів алгоритм розв'язання задачі (3.1)–(3.3) буде складатися з таких кроків. Узагальнена структурна схема алгоритму зображена на рис. 3.24.

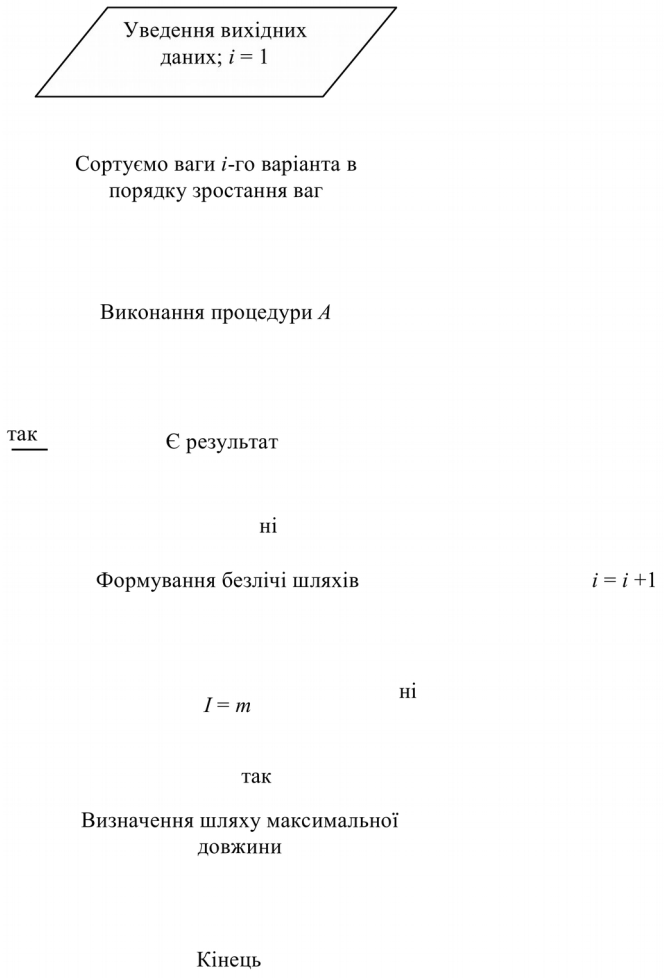


Рис. 3.24. Узагальнена структурна схема алгоритму

Крок 1. $i=1$, сортуємо ваги i-ї множини в порядку зростання.

Крок 2. Виконуємо процедуру A' для поточного значення ваг.

Крок 3. Перевіряємо процедурою A', отримано кінцевий результат чи ні; якщо так, то алгоритм закінчує роботу, якщо ні, то виконуємо наступний крок.

Крок 4. Формуємо множину $\{\mu_s^{*r}(i)\}$; перевіряємо i=m, якщо $i \neq m$, то $i=i+1$ і переходимо до кроку 2, в іншому випадку виконуємо наступний крок.

Крок 5. У множині $\{\mu_s^{*r}(\text{i})\}$ виділяємо шлях максимально можливого рангу й на цьому алгоритм закінчує роботу.

Приклади розв'язання задачі за допомогою даного алгоритму (рис. 3.24) й алгоритмів різних стратегій, що використовують відсікання, розглянуті в роботах [73–100]. Аналогічно може бути розв'язана й задача мінімізації, функціонала (3.7) при обмеженнях (3.8). У цьому випадку коефіцієнти функціонала необхідно відсортувати в порядку зростання й у структурній схемі алгоритму всі операції знаходження найдовшого шляху по вазі C_j замінити на операції визначення найкоротшого шляху по вазі C_j.

3.5. Ранговий підхід до розв'язання задачі про найменше покриття й розбиття

3.5.1. Формальна модель ЗНП

Нехай A^m – транспонована матриця суміжності графа $D\Delta$ з одиничними діагональними елементами. Задача визначення найменшої домінуючої множини графа G еквівалентна задачі знаходження такої найменшої множини стовпців у матриці A^m, де кожний рядок матриці містить одиницю хоча б в одному з обраних стовпців. Ця задача про пошук найменшої множини стовпців, «покриваючих» одиницями всі рядки, одержала назву ЗНП. У загальному випадку матриця, що складається з 0 і 1, не обов'язково є квадратною. Крім того, кожному стовпцю j (вершині x_j) у матриці A^m зіставляється деяка вага, і потрібно знайти покриття з найменшою загальною вартістю. У випадку ж рівності коефіцієнтів задача трансформується в задачу мінімізації кількості стовпців, що покривають всі рядки в матриці Ат.

Інакше кажучи, необхідно мінімізувати цільову функцію

$$L = \sum_{j=1}^{n} c_j x_j \rightarrow \min \tag{3.38}$$

при обмеженнях

$$\begin{aligned} &\sum_{j=1}^{n} a_{ij} x_j \geq 1, \ \text{i} = \overline{1, m}; \\ &x_j \in \{0,1\}; \ c_j \geq 0; \end{aligned} \tag{3.39}$$

де

$$a_{ij} = \begin{cases} 1 \text{ - коли i - змінна може бути покрита змінною } x_j \\ 0 \text{ - коли i - змінна може не бути покрита змінною } x_j. \end{cases} \tag{3.40}$$

3.5.2. Ранговий підхід до розв'язання ЗНП

Нехай задана булева матриця $B = \| \beta_{ji} \|$ з *n* стовпцями й *m* рядками. Стовпці матриці B будемо задавати *m-мірними* векторами $B^{(i)} = (\beta_1^i, \beta_2^i, ..., \beta_m^i)$, у яких компоненти β_j^i набувають значення, що дорівнює *j*, якщо *j–й* рядок покривається однієї з одиниць *i–го* стовпця, і дорівнює 0 в іншому випадку. Кількість компонентів β_j^i у векторі $B^{(i)}$, що дорівнюють нулю, назвемо ваговою характеристикою γ_i вектора $B^{(i)}$. Геометрично ЗНП, обумовлена співвідношеннями (3.38) – (3.39), інтерпретується як задача відшукання оптимальної вершини одиничного куба в R^n. Побудуємо на основі матриці *B* еквівалент *n-мірного* одиничного куба R^n у вигляді стягнутого дерева усіх шляхів [132, 133], заданого графом *G*Δ (див. рис. 3.4), вершини якого утворять *n* горизонтальних лінійок. Як показано в роботі [110], множини шляхів від фіктивної вершини *s* до усіх інших вершин цього графа відповідає $2n^{-1}$ варіантам можливих розв'язків. Покладемо, що первісно вектори $B^{(i)}$ відсортовані по вагах γ_i у порядку їхнього зростання й пронумеровані так, що заданій послідовності векторів $B^1, B^2, ..., B^n$ відповідає нерівність $\gamma_1 < \gamma_2 < ... < \gamma_n$. Поставимо у відповідність кожній вершині графа *D*Δ вектор $B^{(i)}$, $i = \overline{1,n}$. Шлях $\mu_{st}^{p,l,...,t}$ у графі *D*Δ, що проходить через *k* вершин, будемо розглядати як об'єднання *k* векторів, що відповідають вершинам *p, l...t*, що ввійшли в цей шлях. Тобто якщо в шлях $\mu_{st}^{p,l,...,t}$ увійшли вершини *p, l...t*, то шляху $\mu_{st}^{p,l,...,t}$ відповідає вектор $B^{(p,l...t)} = B^{(p)} \cup B^{(l)} \cup ... \cup B^{(t)}$, при цьому правило об'єднання однойменних компонентів β_j^i, що належать різним векторам $B^{(p)} \cup B^{(l)}$ *(p < l)*, таке:

$$\beta_j^p \cup \beta_j^l = \begin{cases} 0, \text{ если } \beta_j^p = 0; \beta_j^l = 0, \\ \beta_j^p, \text{если } \beta_j^p = \beta_j^l \text{ или } \beta_j^l = 0, \\ \beta_j^l, \text{если } \beta_j^p = 0. \end{cases} \qquad (3.41)$$

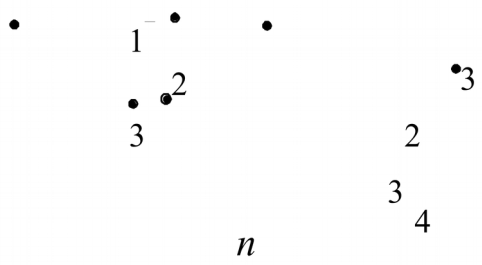


Рис. 3.25. Граф *G*Δ, вершини якого утворять *n* горизонтальних лінійок

Тоді ЗНП можна розглядати як задачу визначення найкоротшого шляху від вершини *s* у графі *D*Δ до усіх інших вершин цього графа по вагах γ_i, що містить найменшу кількість вершин. Покажемо, що введення попередньої розмітки графа *D*Δ на основі каліброваних векторів дозволяє звести ЗНП до задачі визначення НШ у графі *D*Δ від вершини *s* до усіх інших вершин графа. Слід зазначити, що одержання точного розв'язку задачі про найменше покриття на основі визначення НШ в *D*Δ не може бути здійснено традиційними алгоритмами [77] визначення НШ, оскільки НШ в *D*Δ, що відповідає мінімальному покриттю, не задовольняє властивість того, що кожний відрізок цього НШ є НШ між відповідними вершинами в *D*Δ. Відсутність цієї властивості для НШ в *D*Δ і спричиняється NP–повноту [120–122] задачі про найменше покриття. Тому якщо намагатися розв'язувати задачу (3.38)–(3.39) методами визначення НШ, що наводяться в роботі [77], то можна розраховувати тільки наближені розв'язки задачі (3.38)–(3.39).

Уведемо поняття каліброваних векторів $H^{(i)}$ висот $h_j^i,\ i=\overline{(1,n)},\ j=\overline{(1,m)}.$ Формування векторів почнемо з останньої вершини *n* графа *D*Δ. Перший калібрований вектор $H^{(n)}$ одержуємо в такий спосіб. Всі висоти h_j^n, для яких β_j^n в $B^{(n)}$ дорівнюють 0, одержують значення $h_j^n=\infty$, а ті, для яких $\beta_j^n=j$, значення $h_j^n=0$. Наступний калібрований вектор $H^{(n-1)}$ будується на основі попереднього вектора $H^{(n)}$ і поточного значення вектора $B^{(n-1)}$.

У загальному випадку на k–й ітерації процес формування вектора $H^{(k)}$ можна записати у вигляді такої рекурентної функціональної залежності:

$$\mathrm{H}^{(\mathrm{k})}=\mathrm{f}\left(\mathrm{B}^{(\mathrm{k})},\mathrm{H}^{(\mathrm{k}+1)}\right). \tag{3.42}$$

При цьому висоти h_j^k вектора $H^{(k)}$ визначаються значеннями компонентів β_j^k і висот h_j^{k+1} вектора $H^{(k+1)}$ у такий спосіб:

$$h_j^k=\begin{cases}0, & \text{если } \beta_j^k=\mathrm{j}\left(\forall h_j^{(k+1)}\right),\\ h_j^{(k+1)}+1, & \text{если } h_j^{(k+1)}\neq\infty,\ \beta_j^k=0,\\ \infty, & \text{если } h_j^{(k+1)}=\infty,\ \beta_j^k=0.\end{cases} \tag{3.43}$$

На основі співвідношень (3.41) – (3.43) калібровані вектори будуються для усіх вершин графа DΔ, починаючи з вершини *i=n* і закінчуючи вершиною *i=1*, тобто одержимо послідовність каліброваних

векторів $H^{(n)}, H^{(n-1)}, \ldots, H^{(1)} \leftrightarrow n, n-1, \ldots, 1$, що відповідають вершинам *n, (n–1),...,1* у графі DΔ.

Фізичний зміст висоти h_j^i, якщо вона не дорівнює 0 або ∞, полягає в тім, що вона вказує номер вершини $x = h_j^i + i$ в графі *D*Δ, приєднання якої до деякого шляху в графі *D*Δ призводить до покриття рядка *j* одиницею в матриці B. Якщо $h_j^i = \infty$, то це означає, що в графі *D*Δ не існує шляху від вершини *i*, що дозволяє утворювати покриття рядка *j* у матриці B. Випадок рівності $h_j^i = 0$ означає, що вектор $B^{(i)}$ покриває рядок *j* у матриці *B*.

Далі, після калібрування, шлях $\mu_{st}^{p,l,\ldots,t}$ в графі *D*Δ, що проходить через вершини *p, l...t*, будемо ставити у відповідність об'єднання каліброваних векторів цих вершин $H^{(p)} \cup H^{(l)} \cup \ldots \cup H^{(t)}$.

Визначимо правила їхнього об'єднання. Оскільки вершини в *D*Δ пронумеровані в порядку зростання при переході від однієї горизонтальної лінійки графа *D*Δ до іншої, а сам граф орієнтований і ациклічний, то при формуванні шляху, наприклад, що складається з двох вершин *p* і *l*, ми будемо розглядати об'єднання векторів $H^{(p)}\ U\ H^{(l)}$ ($p < l$) і не може мати місце об'єднання $H^{(l)}\ U\ H^{(p)}$. Тобто операція об'єднання тут некомутативна, що відповідає спрямованості шляхів у *D*Δ. При об'єднанні довільної пари векторів $H^{(p)}$ U $H^{(l)}$ ($p < l$) поєднуються попарно його однойменні висоти $h_j^{i=p} \cup h_j^{i=l}$.

Позначимо $h_j^{i=p} = x$, $h_j^{i=l} = y$. Тоді правила об'єднання висот можна записати так:

$$\begin{aligned} &\text{x} \cup \text{y} = \text{y}, \ \text{если x} \neq 0, \\ &\text{x} \cup 0 = 0, \\ &0 \cup \text{y} = 0. \end{aligned} \qquad (3.44)$$

Правило (3.7) для об'єднання каліброваних векторів $H^{(i)}$, тобто формування шляхів в *D*Δ, дозволяє ввести вагову характеристику d_i^n вектора $H^{(i)}$, що дорівнює кількості різних висот h_j^i, не рівних 0, в об'єднаному векторі $H^{(i)}$. Наприклад, у векторі $H^{(i)}$=(10012014) міститься три одиниці, двійка й четвірка, отже вагова характеристика цього вектора $d_i^n = 3$. Фізичний зміст цієї характеристики полягає в тім, що вона прогнозує існування не більше d_i^n вершин у *D*Δ, об'єднання з якими дозволить одержати шлях з об'єднаним вектором $H^{(i)}$=(0,0,0...0); $d_i^n = 0$, тобто шлях, що відповідає покриттю усіх рядків стовпцями, що ввійшли в цей шлях, у матриці B. Якщо ж у векторі є хоча б одна висота $h_j^i = \infty$, то й $d_i^n = \infty$, і це означає, що не існує продовження цього шляху в *D*Δ, яке б відповідало покриттю в матриці *B*.

Отже, вагова характеристика d_i^n об'єднаного вектора деякого шляху μ_{st} у графі $D\Delta$ визначає, яку найбільшу кількість вершин необхідно приєднати до цього шляху, щоб одержати шлях з вагою $d_i^n=0$. Таким чином, якщо для кожної вершини в графі $D\Delta$ побудований калібрований вектор $H^{(i)}$ і для кожного $H^{(i)}$ визначена вагова характеристика d_i^n, то найкоротший шлях по вагах d_i^n у графі $D\Delta$ від вершини s до усіх інших вершин графа $G\Delta$ визначає мінімальне покриття матриці B стовпцями, що ввійшли в цей шлях.

Твердження 3.6. Якщо в DΔ побудований шлях $\mu_{st}^{p,l,...,t}$, що проходить через вершини *p, l,...,t*, довжина якого по вагах d_i^n=0, то йому відповідає об'єднаний калібрований вектор $H^{(p,l,...,t)}=\left(h_1^{p,l,...,t}=0, h_2^{p,l,...,t}=0,...,h_n^{p,l,...,t}=0\right); d^{\text{п}}_{\text{p,l,...,t}}=0$, а вершини, що ввійшли в цей шлях, утворять покриття матриці *B*.

Доведення. Рівність $h_j^{p,l,...,t}=0$ означає, що $\beta_j^{p,l,...,t}=j$, що відповідає об'єднанню векторів $B^{(p,l,...,t)}$ по даному шляху і має вигляд

$$\left(\beta_1^{p,l,...,t}=1, \beta_2^{p,l,...,t}=2,..., \beta_m^{p,l,...,t}=m\right),$$

тобто дає покриття усіх рядків одиницями стовпців, що ввійшли в шлях, у матриці B, що й було потрібно довести.

Отже, якщо побудувати деяку процедуру A, що дозволяє ідентифікувати найкоротший шлях у вершино зваженому графі DΔ по вагах $d_i^{\text{п}}$, то цей шлях буде відповідати об'єднанню векторів $B^{(i)}$, що утворять мінімальне покриття в матриці B. Для зручності опису такої процедури в графі DΔ уведена фіктивна вершина *s*, завдяки якій ранг шляху (кількість ребер, що утворять шлях) збігся з кількістю різних вершин у графі DΔ. Розглянемо варіанти побудови правил відсікання $\{L_w\}$ для розв'язання ЗНП.

3.5.3. Правила відсікання безперспективних варіантів при розв'язанні ЗНП на основі рангового підходу

Почнемо розгляд правил відсікання шляхів у множинах m_{si}^r з варіанта незваженої задачі про найменше покриття, коли ваги у функціоналі (3.38) $c_1 = c_2 =...= c_n = 1$. Найбільш простим і природним правилом відсікання шляхів L_1 у множинах $m_{si}^r, i=\overline{(r,n)}$ є видалення шляхів $\mu_{si}^r \in m_{si}^r$, у яких $d_i^n = \infty$, оскільки ці шляхи не задовольняють властивість ν. Надалі ми будемо використовувати поняття песимістичного гарантованого прогнозу d_i^n кількості вершин, приєднання яких до шляху μ_{si}^r дозволить

одержати шлях з вектором $H^{(i)}(0,0,...,0)$, тобто шлях, що відповідає покриттю в матриці B, і оптимістичного негарантованого прогнозу d_i^o, визначення якому буде дано нижче. Кожна вершина в $D\Delta$ характеризується двома числовими величинами: кількістю непокритих рядків у векторі $H^{(i)}$ $ai = m - p_i$, де p_i дорівнює кількості нулів в $H^{(i)}$, і числом $\alpha_i = m - \gamma_i$, що визначає максимально можливу кількість рядків, що може бути покрито при приєднанні до шляху $\mu_{sp}^{r\ r}$ вершини i, тобто ребра (p,i). Причому множина рядків, що покриваються вершиною i, може перетинатися з множиною рядків, покритих вершинами, що вже ввійшли у сформований шлях μ_{sp}^{r}. Тому в новому векторі $H^{(i)}$, що відповідає шляху $\mu_{si}^{r+1} = \mu_{sp}^{r} \cup (p,i)$, кількість нулів у найкращому разі, якщо не відбудеться перетинань, може збільшитися на величину α_i.

Назвемо найменшу кількість вершин d_i^o, приєднання яких до шляху μ_{si}^{r} задовольняє нерівність

$$\sum_{j=1}^{d_i^o} \alpha_{i+j} \geq a_i^H, \tag{3.45}$$

оптимістичним негарантованим прогнозом, що визначає мінімальну кількість вершин d_i^o, що необхідно приєднати до шляху μ_{si}^{r} для того, щоб одержати шлях μ_{st}^{r*}, що відповідає покриттю в матриці B. Вершини i відсортовані в графі DΔ у порядку зростання γ_i і, отже, у порядку зменшення α_i. Уведемо для кожної вершини i допоміжний вектор $z_i(z_{i+1},z_{i+2},...,z_{i+k})$, де $i + k \leq n$, а компоненти вектора визначаються такими рекурентними співвідношеннями:

$$\begin{aligned} z_{i+1} &= \alpha_{i+1}, \\ z_{i+2} &= z_{i+1} + \alpha_{i+2}, \\ &\cdots\cdots\cdots\cdots\cdots\cdots \\ z_{i+k} &= z_{i+(k-1)} + \alpha_{i+k}. \end{aligned} \tag{3.46}$$

Допоміжний вектор z_i для вершини i графа $D\Delta$ дозволяє визначити оптимістичний негарантований прогноз для довільного шляху μ_{si}^{r}, що характеризується вектором $H^{(i)}$. Для цього визначається величина $ai = m - p_i$, і тоді відповідно до виразу (3.9)

$$d_i^o = j, \tag{3.47}$$

де j – значення, починаючи з якого виконується нерівність

$$z_{i+j} \geq a_i^H;\ \mathrm{j} = \overline{1,\mathrm{k}};(\mathrm{i}+\mathrm{j} \leq \mathrm{n}). \qquad (3.48)$$

Якщо величина песимістичної оцінки шляхи μ_{si}^r $d_i^n = \infty$, то й $d_i^o = \infty$. Уведемо поняття коридору в множині шляхів μ_{si}^r на r ярусі графа $D\Delta$ $\left(i = \overline{r,n}\right)$. Під виділенням коридору будемо розуміти процес відсікання шляхів у множині μ_{si}^r відповідно до правила p.

Правило p

Шлях μ_{sk}^r можна виключити з подальшого аналізу, якщо виконується нерівність

$$d_k^o\left(\mu_{sk}^r\right) > d_i^{*n}\left(\mu_{si}^r\right), \qquad (3.49)$$

де d_i^{*n} – кращий на даний момент песимістичний прогноз на ранзі r.

Теорема 3.6. Використання правила p у процесі розв'язання ЗНП на основі рангового підходу дозволяє виключати з подальшого аналізу безперспективні шляхи, не втрачаючи при цьому оптимального розв'язку задачі.

Доведення. Припустимо, що, використовуючи правило p, ми відсікли шлях, що надалі дав би оптимальний розв'язок задачі на ранзі $r\tilde{}=r+x$. Але в цьому випадку виходить, що $x < d^{*n}_i < d_k^o$, тобто ми одержимо покриття за $x < d_k^o$ кроків, що суперечить визначенню оптимістичного прогнозу, що й було потрібно довести.

У принципі, доведення цієї теореми очевидно, тому що ми відсікаємо лише ті шляхи, оптимістичний прогноз яких гірше, ніж кращий, існуючий на даний момент песимістичний. А це означає, що виключений шлях μ_{sk}^r у найкращому разі може дати покриття за $r+d_k^o$ кроків, що явно більше значення $r + d^{*n}_i$, що ми одержимо в найгіршому разі, використовуючи шлях μ_{si}^r, песимістичний прогноз якого є на даний момент найкращим.

Таким чином, розглянуті нами правила відсікання безперспективних шляхів μ_{sk}^r у множинах m_{sk}^r $(\mathrm{k} = \overline{\mathrm{r,n}})$ L_1 і p можуть бути використані для одержання точного розв'язку ЗНП, якщо їх застосувати в процедурі A_0.

Для зменшення часової складності розв'язання задачі (3.38)– (3.39) можна використовувати процедуру відсікання L_2, обумовлену рекурентним співвідношенням

$$\forall\left(\mu_{si}^{r} \in m_{si}^{r}\right): \begin{cases} \mu_{sp}^{r+1} = \min\limits_{d_i^n} \left\{ \mu_{si}^{r} \cup (i,p) \right\}, \\ p = \overline{r+1, n};\ i = \overline{r, n}, \end{cases} \tag{3.50}$$

що характеризує процес формування шляхів наступного рангу з урахуванням принципу оптимізації по напрямку. Застосування правила (3.20) дозволяє відсікати більшу кількість шляхів, але при цьому є ймовірність втрати оптимального розв'язку.

Розглянемо тепер випадок, коли c_i у функціоналі (3.38) приймають довільні значення. У цьому випадку змінні у виразі (3.38) необхідно перенумерувати так, щоб виконувалася нерівність

$$c_1 \le c_2 \le \ldots \le c_n . \tag{3.51}$$

У зв'язку з тим, що кожному стовпцю в матриці В тепер відповідає вага c_i, виникає необхідність у визначенні оцінок довжини шляху μ_{si}^{r} з урахуванням ваг вершин: оптимістичної негарантованої $d_i^{ов}$ і песимістичної $d_i^{пв}$.

Для одержання оптимістичної оцінки $d_i^{ов}$ ми не можемо скористатися співвідношеннями (3.45)–(3.48), оскільки характеристика α_i при зміні i змінюється через сортування нерівності (3.51). У зв'язку з цим для визначення характеристик $d_i^{ов}$ і $d_i^{пв}$ уведемо систему каліброваних векторів $\{y_i\}$ і $\{z_i\}$. Для цього складемо вектор $a\left(a_1^{\square}, a_2^{\square}, \ldots, a_n^{\square}\right)$, у якому $a_i^{\square}$ розташовані в порядку зменшення:

$$a_1^{\square} \ge a_2^{\square} \ge \ldots \ge a_n^{\square} . \tag{3.52}$$

Якщо позначити a_q` = α_{iq}, де $q = \overline{1, n}$, а i – відповідає номеру вершини i у графі $D\Delta$, для якої визначалося a_i, то справедливою є рівність

$$a = \alpha = \left(\alpha_{i_1 1}, \alpha_{i_2 2}, \ldots, \alpha_{i_n n}\right). \tag{3.53}$$

Використовуючи вектор α, сформуємо множину векторів $\{y_i\}$. Для одержання вектора y_1 у векторі α викреслюємо елемент α_{1l} і для усіх α_{iq}, у яких $q > l$, значення другого індеКЗу q зменшуємо на 1. Після цього формуємо компоненти $\{y_{1k}\}$ вектора y_1 за правилом

$$y_{1k} = \alpha_{1k} + y_{1(k-1)};\ k = \overline{0, n-1};\ y_{10} = 0 . \tag{3.54}$$

Далі у векторі α викреслюємо елемент α_{2j}, змінюючи по q нумерацію так само, як і раніше. За правилом (3.17) продовжуємо формувати компоненти вектора y_2 і т. д. На p–*му* кроці викреслюємо в α елемент α_{pj}, змінюючи, як і раніше, нумерацію по q. Тоді

$$y_{pk} = \alpha_{pk} + y_{p(k-1)}; \; k = \overline{0, n-p}. \tag{3.55}$$

Для $p = n$ вектор y_n буде мати тільки один елемент, рівний $\alpha_{nq} = \alpha_n$, де q відповідає номеру цього елемента у векторі α.

Формування величини z_p^o, що є оптимістичним прогнозом ваги вершин, які необхідно приєднати до шляху μ_{sp}^r, щоб можливо було одержати покриття, здійснюється на основі такого співвідношення:

$$z_i^o = \sum_{j=p+1}^{y_{ik^*}+p+1} c_j, \tag{3.56}$$

де y_{ik^*} – елемент у векторі y_i, починаючи з якого виконується нерівність $y_{ik^*} > \gamma_i^{(H)}$.

За допомогою вектора y_i і співвідношення (3.56) легко здійснити негарантований оптимістичний прогноз мінімальної кількості вершин, які необхідно додати до шляху μ_{si}^r в графі $D\Delta$, щоб одержати покриття в матриці В. Сформулюємо правило k_1 визначення $d_i^{ов}$ і $d_i^{пв}$.

Правило k_1:

а) обчислюємо для шляху μ_{si}^r у вершину i величину $\gamma_i = m - pi$, де pi – кількість шляхів у векторі $H^{(i)}$, що відповідає цьому шляху;

б) у каліброваному векторі y_i для вершини i знаходимо номер k^*, починаючи з якого $y_{ik^*} > \gamma_i^{(H)}$. Значення $y_{ik^*} = d_i^o$ дає незважений негарантований оптимістичний прогноз;

в) визначаємо зважений негарантований оптимістичний прогноз

$$d_i^{ов} = \sum_{j \in \mu_{si}^r} c_j + z_i^o; \tag{3.57}$$

г) величина зваженого песимістичного прогнозу визначається за формулою

$$d_i^{пв} = \sum_{j \in \mu_{si}^r} c_j + \sum_{j=n-d_i^n+1}^{n} c_j. \tag{3.58}$$

Правило виділення коридору $p^{в}$ у випадку зважених стовпців у матриці B при розв'язанні задачі (3.38)–(3.39) аналогічно виразу (3.49).

Правило $p^{в}$

Якщо позначити через d_l^{*nb} величину найменшого песимістичного прогнозу деякого шляху μ_{sl}^r, тоді всі шляхи, для яких виконується нерівність

$$d_k^{ов}\left(\mu_{sk}^r\right) > d_l^{*нв}\left(\mu_{sl}^r\right), \tag{3.59}$$

можна видалити з подальшого аналізу як безперспективні. Правило L_2, обумовлене співвідношенням (3.50), також залишається колишнім, але при цьому мінімум береться по $d_i^{нв}$.

$$\forall\left[\mu_{si}^r \in m_{si}^r\right]:\left[\mu_{sp}^{r+1} = \min_{d_i^{нв}}\left(\mu_{si}^r \cup (i,p)\right)\right], \tag{3.60}$$
$$p=\overline{r+1,n};\ i=\overline{r,n}.$$

3.5.4. Наближені й точні алгоритми розв'язання ЗНП

Розгляд алгоритмів розв'язання ЗНП почнемо з наближеного алгоритму розв'язання задачі (3.38)–(3.39) для випадку рівності коефіцієнтів c_i з використанням правил об'єднання L_1 і L_2. Тоді ЗНП можна розглядати як задачу визначення найкоротших шляхів на основі рангового підходу [110] у графі $D\Delta$ з урахуванням принципу оптимізації по напрямку (3.48). Покроковий опис алгоритму визначення найкоротших шляхів у графі $D\Delta$, що задовольняють властивість ν, наведено нижче.

Наближений алгоритм розв'язання незваженої ЗНП (процедура $A_к$)

Крок 1. Формуємо калібровані вектори $H^{(i)}$, виділяємо множини шляхів $\{\mu_{si}^{r=1}\}$, що задовольняють властивість ν. Якщо всі ці множини порожні, то алгоритм закінчує роботу, тому що це означає, що покриття не існує. Якщо ж є хоча б одна непорожня множина, то визначаємо в кожній з них шляхи $\mu^*_{si}{}^{r=1}$, мінімальні по вазі d_i^n, і робимо перехід до кроку 3.

Крок 2. На основі шляху множини шляхів $\{\mu^*_{si}{}^r\}$ поточного рангу формуємо множини шляхів μ_{si}^{r+1} наступного рангу, що задовольняють властивість ν, виділяємо в них шляхи $\mu^*_{si}{}^{r+1}$ з найменшим значенням d_i^n (правило L_2) і переходимо до виконання наступного кроку.

Крок 3. Якщо є $d_i^n = 0$, то алгоритм закінчує роботу, інакше – виконуємо крок 4.

Крок 4. Якщо є $d_i^n = 1$, то формуємо шлях наступного рангу з $d_i^n = 0$ і алгоритм закінчує роботу, інакше – значення r збільшуємо на 1 і переходимо до кроку 2.

Наведений алгоритм відображає основне функціональне рівняння динамічного програмування, що для даної задачі можна подати як

$$d_{\min}^{r=k}(i) = \min_i \left[d_{\min}^{r=k-1}(s,j) + d^{r=k}(j,i) \right], \qquad (3.61)$$

де $d_{\min}^{r=k-1}(s,j)$ – умовна мінімальна відстань від вершини s до вершини j в DΔ, знайдена на (k–*1*) кроці роботи процедури;

$d^{r=k}(j,i)$ – величини ваг вершин, що утворять шляхи до вершин $i=(1,n)$ на k–*му* кроці.

Рівняння (3.61) дозволяє по ланцюжку знаходити умовні оптимальні розв'язки.

Таким чином, визначення каліброваних векторів $H^{(i)}$ еквівалентно маркуванню шляхів у графі $D\Delta$ від кінця до початку, а використання потім процедури $A_к$ є нічим іншим, як ідентифікацією найкоротшого шляху по зробленому маркуванню на основі функціонального рівняння динамічного програмування (3.61), коли процес програмування розгортається від початку до кінця. У випадку зважених стовпців у матриці В покроковий опис наближеного алгоритму $A^в_k$ розв'язання задачі (3.38)–(3.39) з використанням правила відсікання L_2 залишається тим самим, але при цьому пошук найкоротших шляхів іде за ваговою характеристикою $d_i^{нв}$, обумовленою співвідношенням (3.61).

Наближений алгоритм розв'язання зваженої ЗНП (процедура $A^в_к$)

Крок 1. Формуємо калібровані вектори $H^{(i)}$, виділяємо множини шляхів $\{\mu_{si}^{r=1}\}$, що задовольняють властивість ν. Якщо всі ці множини порожні, то алгоритм закінчує роботу, тому що це означає, що покриття не існує. Якщо ж є хоча б одна непорожня множина, то визначаємо в кожній з них шляхи $\mu^*_{si}{}^{r=1}$, мінімальні по вазі $d_i^{нв}$, і переходимо до кроку 2.

Крок 2. Якщо є шляхи з $d_i^{нв} = 0$, то виділяємо серед них шлях, мінімальний по вазі, що визначає локальний екстремум на ранзі, і переходимо до наступного кроку.

Крок 3. На основі шляху множини шляхів $\{\mu^*_{si}{}^r\}$ поточного рангу r формуємо множини шляхів μ_{si}^{r+1} наступного рангу відповідно до правила L_2 і переходимо до виконання наступного кроку.

Крок 4. Якщо всі множини наступного рангу порожні, то виділяємо з множини локальних екстремумів глобальний і алгоритм закінчує роботу. Якщо ж є хоча б одна непуста множина, то збільшуємо значення поточного рангу $r:=r+1$ і переходимо до кроку 2.

Точне розв'язання задачі (3.38)–(3.39) у випадку рівності коефіцієнтів c_i можливо тільки з використанням правил L_1 і p. При цьому покроковий опис алгоритму, реалізованого на основі процедури A_0, наведено нижче.

Точний алгоритм розв'язання незваженої ЗНП (процедура А)

Крок 1. Формуємо калібровані вектори $H^{(i)}$, виділяємо множини шляхів $\{\mu_{si}^{r=1}\}$, що задовольняють властивість ν. Якщо всі ці множини порожні, то алгоритм закінчує роботу, тому що це означає, що покриття не існує. Якщо ж є хоча б одна непуста множина, то для усіх вершин графа $G\Delta$ обчислюємо допоміжні вектори z_i, визначаємо для кожного шляху характеристики d_i^n і d_i^o відповідно до співвідношень (3.55)–(3.58) і переходимо до виконання наступного кроку.

Крок 2. На основі шляху множини шляхів $\{\mu^*_{si}{}^r\}$ поточного рангу r формуємо множини шляхів μ_{si}^{r+1} наступного рангу r+1 на основі рекурентного співвідношення (3.47), визначаємо характеристики цих шляхів d_i^n і d_i^o відповідно до виразів (3.55)–(3.58) і виділяємо коридор у множині m_{si}^{r+1} за правилом p, обумовленим співвідношенням (3.59).

Крок 3. Якщо є $d_i^n = 0$, то алгоритм закінчує роботу, тому що саме цей шлях утворить мінімальне покриття, інакше – виконуємо крок 4.

Крок 4. Якщо є $d_i^n = 1$, то формуємо шлях наступного рангу з $d_i^n = 0$ і алгоритм закінчує роботу, інакше – значення r збільшуємо на 1 і переходимо до кроку 2.

У випадку, коли стовпці в матриці В зважені, покроковий опис алгоритму розв'язання задачі (3.38–3.39) набуває такого вигляду, як наведено нижче.

Точний алгоритм розв'язання зваженої ЗНП (процедура $A^в$)

Крок 1. Формуємо калібровані вектори $H^{(i)}$, виділяємо множини шляхів $\{\mu_{si}^{r=1}\}$, що задовольняють властивість ν. Якщо всі ці множини порожні, то алгоритм закінчує роботу, тому що це означає, що покриття не існує. Якщо ж є хоча б одна непуста множина, то для усіх вершин графа $D\Delta$ визначаємо калібровані вектори y_i і z_i, обумовлені співвідношеннями (3.52)–(3.56), визначаємо для кожного шляху характеристики $d_i^{nв}$ і $d_i^{oв}$ відповідно до співвідношень (3.57)–(3.58), виділяємо в них шляхи $\mu^*_{si}{}^{r=1}$ відповідно до правил L_1 і $p^в$ і переходимо до виконання наступного кроку.

Крок 2. Якщо є шляхи з $d_i^{nв} = 0$, то виділяємо серед них шлях, мінімальний по вазі, що визначає локальний екстремум на ранзі, і переходимо до наступного кроку.

Крок 3. На основі шляху множини шляхів $\{\mu^*_{si}{}^r\}$ поточного рангу формуємо множини шляхів μ_{si}^{r+1} наступного рангу на основі рекурентного співвідношення (3.10), визначаємо характеристики цих шляхів $d_i^{nв}$ і $d_i^{oв}$ відповідно до виразів (3.20)–(3.21) і виділяємо коридор у множині m_{si}^{r+1} за

правилом $p^в$, обумовленим співвідношенням (3.22), і переходимо до виконання наступного кроку.

Крок 4. Якщо всі множини наступного рангу порожні, то виділяємо з множини локальних екстремумів глобальний і алгоритм закінчує роботу. Якщо ж є хоча б одна непуста множина, то збільшуємо значення поточного рангу *r:=r+1* і переходимо до кроку 2. Таким чином, показана можливість застосування методу розв'язання оптимізаційних задач із БЗ на основі рангового підходу до розв'язання ЗНП. Оскільки ЗНП поліноміально зводиться до таких задач [162]: найменша домінуюча множина вершин у графі; найбільша незалежна множина вершин у графі; найбільша кліка в графі; найбільше сполучення в графі, то розглянуті наближені й точні алгоритми розв'язання ЗНП можуть бути використані й для розв'язання перерахованих задач, що мають широке прикладне значення в теорії складних систем [177].

3.5.5. Особливості рангового підходу при розв'язанні ЗНР

Неважко показати, що якщо в алгоритмах, розроблених для розв'язання ЗНП, правила об'єднання векторів (3.44) замінити на вирази

$$\begin{aligned} x \cup y &= y, \text{ если } x \neq 0, \\ x \cup 0 &= 0, \\ 0 \cup y &= 0, \\ 0 \cup 0 &= \infty, \end{aligned} \qquad (3.62)$$

то їх можна застосувати для розв'язання ЗНР.

У роботах [99, 105, 110–112] показано, що при використанні рангового підходу в наближених алгоритмах на похибку розв'язання істотно можуть впливати сортування векторів n–мірного одиничного куба в графі $D\Delta$. У зв'язку з цим для зважених ЗНП і ЗНР уведемо коефіцієнти: χ_j визначальна кількість одиниць у векторі $B^{(i)}$ і $\lambda_j = c_j / \chi_j$, що характеризує частину вагової характеристики C_j, яка припадає на кожну одиницю вектора $B^{(i)}$. Тоді якщо перед роботою алгоритму застосувати замість сортування виразу (3.15) сортування векторів $B^{(i)}$ у порядку зменшення

$$\lambda_1 > \lambda_2 > \ldots > \lambda_{n-1} > \lambda_n,$$

то становить інтерес проведення дослідження впливу різних сортувань на похибку розв'язання зважених задач ЗНП і ЗНР.

При розв'язанні ЗНР на деякій множині $A=\{a_i\}$ на основі уведеної узагальненої процедури A_0 виникає запитання про можливість розв'язання задачі ЗНР, тобто про існування покриття з непересічних елементів заданої

потужності m. Умова існування розв'язку ЗНР можна сформулювати у вигляді наступної теореми.

Теорема 3.7. ЗНР має рішення тоді й тільки тоді, коли підмножина A={a_i} є розбиттям числа m.

Доведення. Припустимо, що в A немає {a_i} відповідного розбиття m і побудована підмножина $L=\bigcup_{i=1}^{z} l_{si}^{r=1}$ є покриття P потужності m, тоді якщо для $\forall l_{si}^{r=1}\in L$ справедливе рівність $\bigcap_{i=1}^{z} l_{si}^{r=1}=\varnothing$, то з огляду на те, що $l_{si}^{r=1}=a_i$, повинна виконатися й рівність $\sum_i a_i=m$, але це суперечить первісному припущенню про відсутність розбиття числа m і, отже, L у цьому випадку може бути покриттям тоді й тільки тоді, коли виконаються умови теореми.

Наслідок. Якщо $a_1=a_2=\ldots a_q=a$, то ЗНР має розв'язок тільки в тому випадку, якщо ***m*** ділиться на ***a*** без остачі.

Доведення. Якщо деякий шлях μ_{sv} графа G_Δ рангу r задовольняє умову $\bigcap_{i=1}^{r} l_{si}^{r=1}=\varnothing$, де $l_{si}^{r=1}\in\mu_{sv}$, то зрозуміло, що довжина цього шляху буде $d(\mu_{sv})$=r∗a. Нам необхідно знайти шлях μ^*_{st} довжиною $d(\mu^*_{st})=\min_r\{r*a\}, де\ r=(\overline{1,n})$, що еквівалентно знаходженню такого $r=r^*$, при якому $r^* a=m$, звідки $r^*=m/a$, але ранг шляху завжди є цілим числом, що й було потрібно довести.

У випадку коли a_i різні, визначити чи є розв'язок, ми зможемо тільки після закінчення роботи алгоритму на основі процедури A_0. Розв'язок буде відсутній тільки в тому випадку, коли всі побудовані підмножини шляхів за допомогою процедури A_0 і вироблених правил відсікання безперспективних варіантів виявляться порожніми.

3.5.6. Ранговий підхід до розв'язання діофантових рівнянь із БЗ

Для розв'язання діофантових рівнянь із булевими змінними

$$\sum_{i=1}^{n}\alpha_i x_i=b \tag{3.63}$$

і систем діофантових рівнянь

$$\sum_{i=1}^{n}\alpha_{ij} x_i=b_j,\quad j=(\overline{1,m}) \tag{3.64}$$

відомі алгоритми, що здійснюють тільки повний перебір варіантів на бінарному дереві розв'язків [120, 121], хоча розв'язання рівнянь (3.63)–(3.64) має важливе значення в теорії складання розкладів і в теорії розпізнавання образів.

У літературі ці задачі відомі як задачі про камені [187]. Дано n каменів із цілими вагами a_1, a_2, ..., a_n, потрібно вибрати деяку кількість із них так, щоб сума їхніх ваг дорівнювала даному числу b. Уведемо двійкові змінні $x_1, x_2, ..., x_n$, тоді розглянута задача – це рівняння

$$a_1x_1 + a_2x_2 + ... + a_nx_n = b. \qquad (3.65)$$

У задачі про камені, описуваній співвідношенням (3.65), або немає розв'язку, або розмірність розв'язку менше від розмірності опису розглянутого варіанта. Можливе й формулювання багатомірної задачі про камені. Нехай камені мають кілька параметрів ваги $a_1^1, a_2^1, ..., a_n^1$, об'єму $a_1^2, a_2^2, ..., a_n^2$ й ціни в різні часи $a_1^j, a_2^j, ..., a_n^j$, потрібно знайти значення двійкових змінних x_1, x_2, ..., x_n, щоб виконувалася рівність

$$a_1^j x_1 + a_2^j x_2 + ... + a_n^j x_n = b_j \quad j = (1,2,...,m). \qquad (3.66)$$

Значення a_i^j й b_j – цілі числа, звичайно не додатні. Часто розглядають задачу з нерівностями, які, як відомо, за рахунок збільшення розмірності задачі зводяться до задачі з рівностями. У роботі [187] показано, що задача про камені належить до класу РС, тому розглянемо можливості побудови точного алгоритму розв'язання задачі на основі рангового підходу, уводячи відсікання безперспективних варіантів на основі поняття коридору. За аналогією з задачею лінійного програмування з булевими змінними зведемо задачу розв'язання систем діофантових рівнянь до задачі визначення шляхів заданої довжини в деякому ациклічному графі $D\Delta$. Ребрам, що стоять у вершини графа $D\Delta$, що відповідають змінним x_1, x_2, ..., x_n, привласнимо ваги a_i^j, що стоять при x_i у рівнянні. Якщо ми будемо розглядати систему рівнянь, то ребрам, що входять у вершини 1, 2 ... n, будемо привласнювати кілька ваг a_i^j. Тоді задача розв'язання рівняння (3.65) зводиться до визначення в ациклічному графі $D\Delta$ шляхів довжини b. Якщо розглядати систему рівнянь (3.64), то в графі $D\Delta$ ребру відповідає m ваг, і для її розв'язання потрібно знайти такий шлях, щоб його довжина по усіх вагах дорівнювала відповідно b_j (j = 1, 2, …, m). Алгоритм розв'язання даної задачі має такий вигляд, як наведено нижче.

Алгоритм розв'язання діофантового рівняння з булевими змінними

Крок 1. З вершини S формуються множини шляхів m_{sj}^{r} рангу $r = 1$, $j=(1\div n)$, довжини яких $d\mu_{sj}^{r} \le b$. Визначаємо коефіцієнти:

$$\gamma_j = c_{j+1}+c_{j+2}+...+c_n\,. \tag{3.66}$$

Крок 2. Формуються множини шляхів $m_{sj}^{r=r+1}$ ($j = r+1, n$) наступного рангу $r = r + 1$ на основі множини шляхів m_{sj}^{r} попереднього рангу r у відповідності з рекурентним співвідношенням

$$\mu_{sj}^{r=r+1} = \{m_{sj} \cup (i, j)\},\ d\mu_{sj}^{r} \le b,\ j = (r\div n),\ i = (r\div n),\ j \ne i. \tag{3.67}$$

При цьому враховується умова

$$d(\mu_{sj}^{r=r+1}) + \gamma_j < b. \tag{3.68}$$

І якщо вона не виконується, то шлях виключається з подальшого аналізу як неперспективний.

Крок 3. Перевіряється множина шляхів $m_{sj}^{r=r+1} = \varnothing$. І якщо вона порожня, то алгоритм закінчує роботу, в іншому випадку здійснюється перехід до кроку 2.

Запропонований алгоритм являє собою процедуру перерахування коренів рівняння (3.63) з виділенням коридору на основі співвідношень (3.67), (3.68) і видаленням шляхів, довжини яких $d\mu_{sj}^{r} \le b$. У випадку розв'язання системи рівнянь (3.64) доцільно розв'язувати одне будь-яке рівняння й перевіряти потім підстановкою перерахованих коренів у всі рівняння, що залишилися, сумісність розглянутої системи.

3.6. Експериментальне дослідження алгоритмів розв'язання задачі ЦЛП із БЗ на основі рангового підходу

3.6.1. Показники ефективності алгоритмів

Порівняння розроблених алгоритмів розв'язання задачі 0,1–рюкзак проведено з адитивним алгоритмом Балаша, Н-методом і методом вектора спаду. Алгоритм Балаша на графіках позначений як алгоритм D. Порівняння алгоритмів розв'язання ЗНП проведено з кращими алгоритмами, що реалізують ідеї методу галузей і границь. Порівняння й дослідження властивостей алгоритмів проводилося за такими параметрами:

1. Кількість ЕО додавання й порівняння, виконуваних алгоритмами.
2. Об'єм пам'яті (V) для реалізації конкретного алгоритму.

3. Для оцінки якості наближених алгоритмів обрано абсолютну й відносну похибки.

Для об'єктивного порівняння розроблених алгоритмів з відомими, а також з урахуванням їхньої специфіки додатково будемо використовувати такі показники якості:

1. Кількість елементарних операцій (ЕО) додавання й порівняння, виконуваних алгоритмами.

2. $t_{ср}$ – середній час розв'язання задачі в секундах (цей параметр залежить від обраної мови програмування, технічних даних ЕОМ, кваліфікації програміста).

3. $K_н$ – кількість неточних розв'язків, що припадають на 100 тестових задач.

4. Похибка наближених розвязків Δf, %, дорівнює

$$\Delta f = \frac{\left| f^* - f^{\sim} \right|}{f^*},$$

де f – значення точного розв'язку тестової задачі;

$f^{\sim}$ – значення наближеного розв'язку тестової задачі.

5. $r_{ср}$ – середній ранг шляхів, що відповідають оптимальному розв'язку тестової задачі, обумовлений кількістю одиниць у векторі x.

6. Коефіцієнт, що характеризує якість фільтрації векторів при виділенні m–мірного коридору в множинах на всьому ярусі порівняно з виділенням m–мірного коридору в множинах, дорівнює

$$K_ф = \frac{K(\bigcup_{j=r}^{n} m_{sj}^r)}{K(m_{sj}^r)},$$

де $K(\bigcup_{j=r}^{n} m_{sj}^r)$ – кількість векторів, що залишилися в коридорі, при виділенні його на ярусі;

$K(m_{sj}^r)$ – кількість векторів, що залишаються в коридорі, при виділенні його в множинах m_{sj}^r.

7. K_y=ЕО(П)/ЕО(n) – коефіцієнт прискорення, що може бути отриманий при реалізації розроблених алгоритмів на n процесорах порівняно з послідовними реалізаціями алгоритмів.

Такі показники, як займаний об'єм пам'яті V, Δf, $\bar{r}$, $K_н$ і $K_ф$, утворять групу $\{\Phi_k\}$–показників, що не залежать від часу виконання алгоритму. А часові показники: число $N_{оп}$ і $K_в$ утворять групу $\{\Theta_m\}$–показників.

Використання показників функціональної групи дозволяє оцінити ефективність застосування правил відсікання $\{L_w, K_i\}$ безперспективних варіантів в узагальненій процедурі A_0, а група часових показників дозволяє зрівняти розроблені алгоритми з відомими. Такий підхід широко використовується при експериментальному дослідженні алгоритмів [177–180].

Для перевірки правильності й точності розв'язання розробленими алгоритмами $A_1 - A_{12}$ як еталон обрано модифікований алгоритм Балаша [75], що враховує вимоги додатності коефіцієнтів у функціоналі. Надалі під алгоритмом D будемо розуміти саме цей алгоритм.

У ході розв'язання тестових задач за допомогою датчика випадкових чисел генерувалися коефіцієнти у функціоналі (3.1) у діапазоні [1÷100] і в обмеженнях (3.2) у діапазоні [1÷10]. Вибір інших діапазонів для функціонала змінив лише його абсолютне значення, але в середньому не вплинув на параметри алгоритмів. Зміна діапазону в обмеженнях впливає лише на ранг шляху (ранг шляху – кількість вершин графа $D\Delta$, які утворюють цей шлях).

Структура дослідження кожного алгоритму полягала в наступному. Розв'язувалося 1000 тестових задач із заданими вихідними параметрами m і n обраним алгоритмом. У ході розв'язання обчислювалися показники ефективності алгоритмів. Отримані значення показників зображені на графіках, апроксимованих за методом найменших квадратів.

Всі алгоритми розділимо на три групи. Першу групу становлять одномірні алгоритми A_1, A_3, A_6, що розв'язують задачу (3.10)– (3.12). Другу групу становлять m–мірні алгоритми A_4, A_5, A_7, A_8, що розв'язують задачу (3.7)–(3.9). А третю групу утворюють багатоетапні одномірні й m–мірні алгоритми $A_9, A_{10}, A_{11}, A_{12}$.

Алгоритм A_2 не ввійшов у жодну з груп, тому що він, з одного боку, є одномірним, а з іншого боку, використовується як один з етапів у багатоетапних одномірних алгоритмах. Найбільший інтерес має не сам алгоритм A_2, а властивість стратегії L_2, на основі якої він побудований. Ця властивість використовується в багатоетапних алгоритмах з метою більш ефективного відсікання векторів на першому ранзі. Тому дослідження алгоритму A_2 не проводилося.

3.6.2. Результати експериментального дослідження алгоритмів

Метою експериментального дослідження з'явилося:

1. Практична перевірка правильності висунутих стратегій відсікань $\{L_w\}$ і $\{K_i\}$, заснованих на доведених у розд. 2 теоремах і твердженнях.

2. Оцінка часової складності запропонованих алгоритмів.

3. Одержання числових значень обраних показників якості алгоритмів у середньому.

4. Порівняння показників ефективності розроблених алгоритмів з відомими алгоритмами.

5. Аналіз причин одержання неточних розв'язків наближеними алгоритмами.

Дослідження алгоритмів першої групи. Як виявилося, кількісні значення обраних показників істотно залежать від рангу $\bar{r}$ одержуваного розв'язку, що визначає кількість одиниць в оптимальному розв'язку. Це підтверджують рис. 3.26 і 3.27, на яких подано залежності часового показника N_{on} алгоритмів від $\bar{r}$ при фіксованих значеннях вихідних параметрів n і m=1.

Рис. 3.26. Залежність N_{on} від $\bar{r}$ при n=15

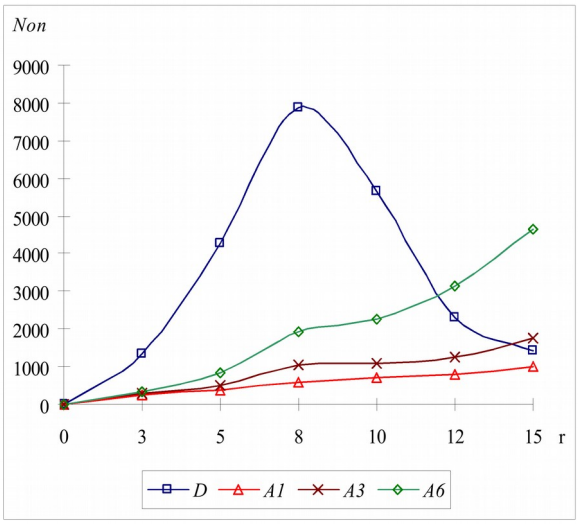


Рис. 3.27. Залежність N_{on} від $\bar{r}$ при n=20

Із рис. 3.26 і 3.27 видно, що діапазон зміни $\bar{r}$ можна умовно розбити на три зони: 1 зона – $\bar{r} = [0 \div n/3]$; 2 зона – $\bar{r} = [n/3 \div 2n/3]$; 3 зона – $\bar{r} = [2n/3 \div n]$. У першій зоні алгоритм D знаходить розв'язок швидко, оскільки за рахунок зондування дуже ефективно відсікаються гілки дерева розв'язків, що відповідають одиничним розгалуженням. Аналогічно пояснюється й швидке одержання розв'язку в 3 зоні, вектори в якій складаються з великої кількості одиниць. Прояв всієї експонентної складності алгоритму D відбувається саме в другій зоні. Для алгоритмів A_1, A_3, A_6 зростання кількості ЕО зі збільшенням $\bar{r}$ визначається кількістю локальних областей W, які необхідно обробити. Згідно з виразом (3.10) кількість таких областей W у графі $D\Delta$ не перевищує $n^{2/2}$, тому при $\bar{r} \to n$ зростання кількості ЕО також не перевищує $c*n2$, де c = const.

Таким чином, для об'єктивного порівняння алгоритмів необхідно вказувати, до якої зони вони належать, тобто який відсоток одиниць (або нулів) містить оптимальний розв'язок. Відповідно до поділу на зони

побудуємо залежність кількості ЕО від n для кожної зони. Ці залежності для зон 1, 2, 3 показані на рис. 3.28, 3.29, 3.30.

Умовимося, що для одержання величини якого-небудь показника для зони будуть використані його усереднені значення на рангах $\bar{r}$, що належать цій зоні.

З наведених рис. 3.28–3.30 видно, що для порівняння часових показників алгоритмів необхідно чітко вказувати приналежність до умовних зон, тому що абсолютне значення цих показників у зонах істотно відрізняється. Потрапляння розв'язку задачі в будь–яку зону визначається співвідношенням діапазону зміни коефіцієнтів a_{1j} в обмеженні (3.11) і заданого діапазону для b_1 у правій частині нерівності (3.11).

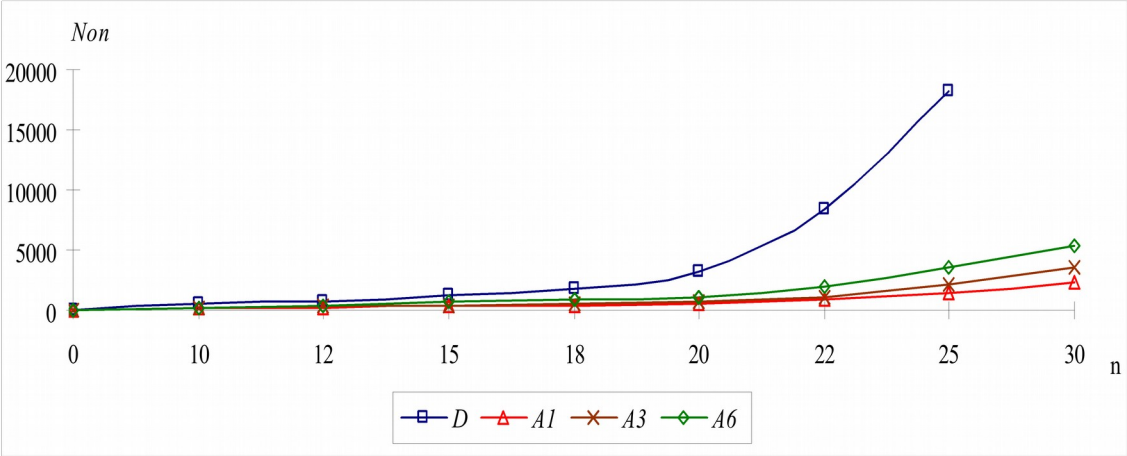


Рис. 3.28. Залежність N_{on} від n для зони 1

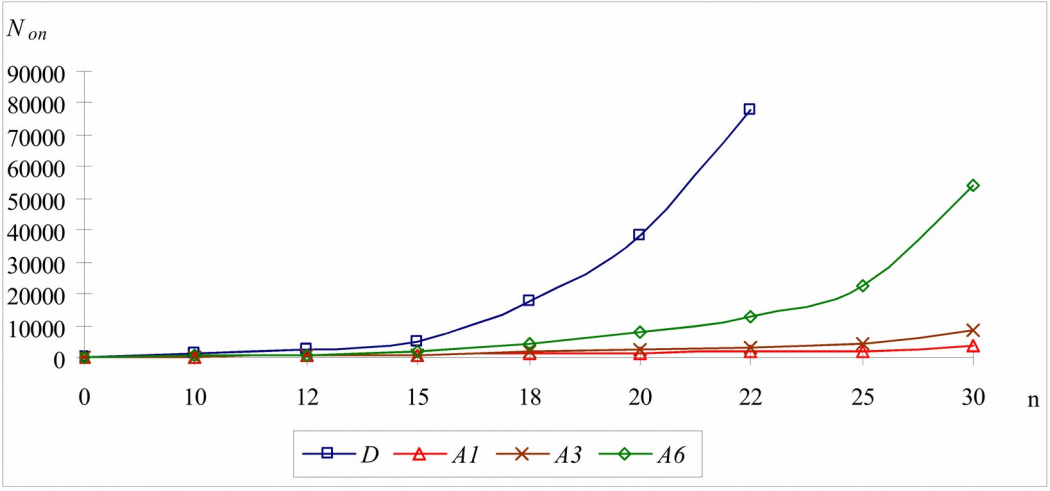


Рис. 3.29. Залежність N_{on} від n для зони 2

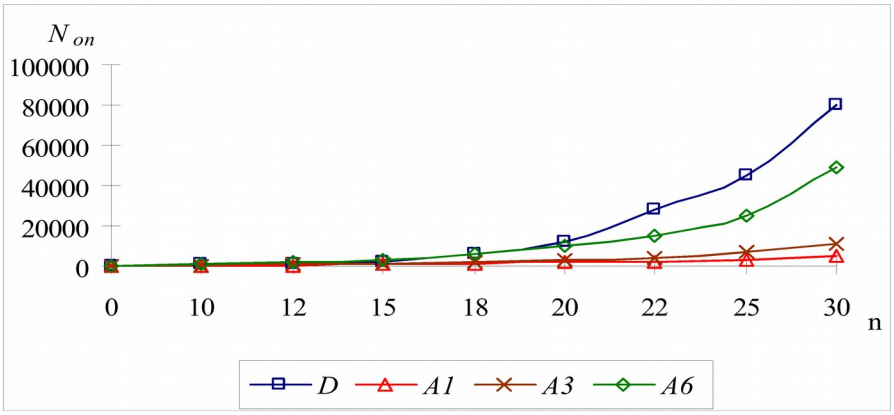


Рис. 3.30. Залежність N_{on} від n для зони 3

Показник необхідного об'єму оперативної пам'яті V залежить від конкретної програмної реалізації. Кількісні значення цього показника, природно, можуть бути змінені й прямо залежать від кваліфікації програміста. Однак вид кривих не зазнає змін через природу тих процесів, які вони описують. Покажемо, що і величина необхідного об'єму оперативної пам'яті для зберігання множини m_{sj}^{r} векторів графа $D\Delta$ також залежить від середнього рангу оптимального розв'язку тестової задачі. Залежність $V = f(\bar{r})$ для алгоритмів D, A_1, A_3, A_6 наводиться на графіках (рис. 3.31 і 3.32).

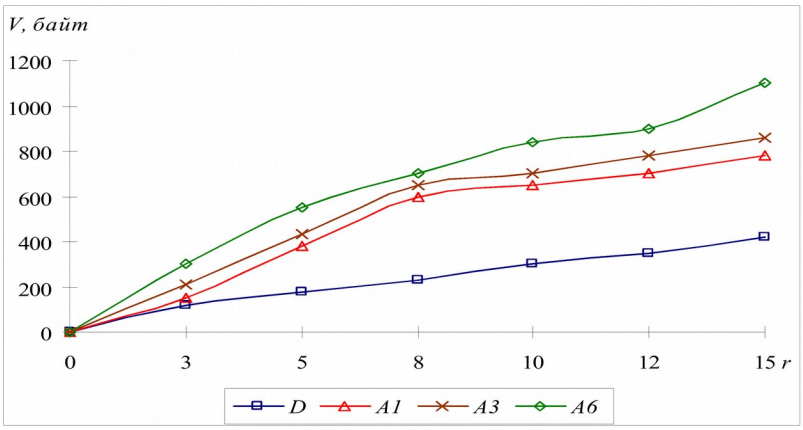


Рис. 3.31. Вплив $\bar{r}$ на V при n=15

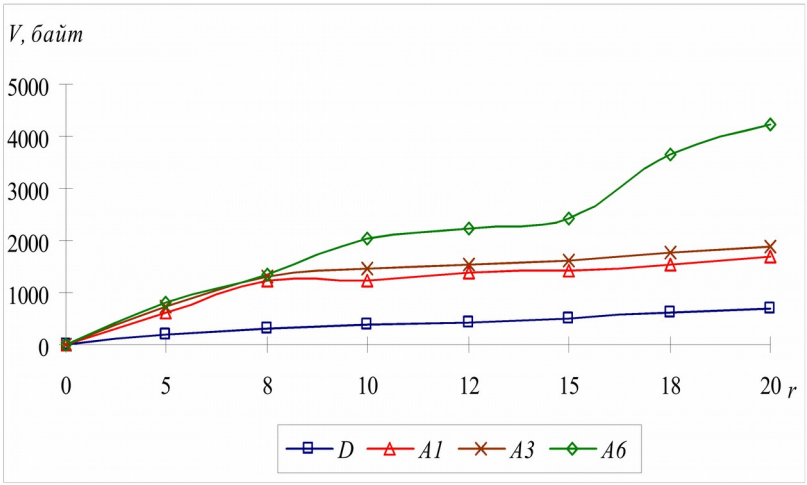


Рис. 3.32. Вплив $\bar{r}$ на V при $n = 20$

Менше значення необхідного об'єму оперативної пам'яті для алгоритму D пояснюється тим, що у випадку успішного зондування по (1) і (0) підмножин альтернатив нова задача, а отже, новий запис у пам'яті не утвориться, тому що коректується список вільних вершин для обраної задачі. Тільки утворення нового розгалуження, що відповідає кроку 6 алгоритму розв'язання діофантового рівняння з булевими змінними, вимагає витрат оперативної пам'яті. Для алгоритмів A_1, A_3, A_6 максимальна кількість оброблюваних векторів перебуває в другій зоні. Тому для зберігання потрібно більше пам'яті, ніж для першої зони. Максимального значення витрати досягають у третій зоні, тому що в цьому випадку відмінність по вагах функціонала усередині множин незначні, що призводить до неефективного відсікання векторів за стратегією L_3.

Залежності величини V від розміру вхідного параметра n також не перевищує $c*n^2$. Результати такого дослідження зображені на рис. 3.33 – 3.35, що відповідають зонам 1 – 3.

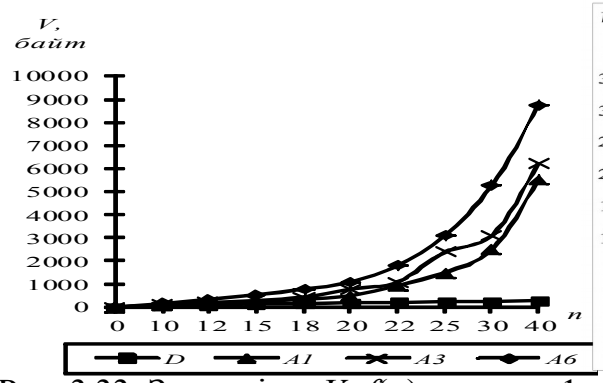


Рис. 3.33. Залежність $V=f(n)$ для зони 1

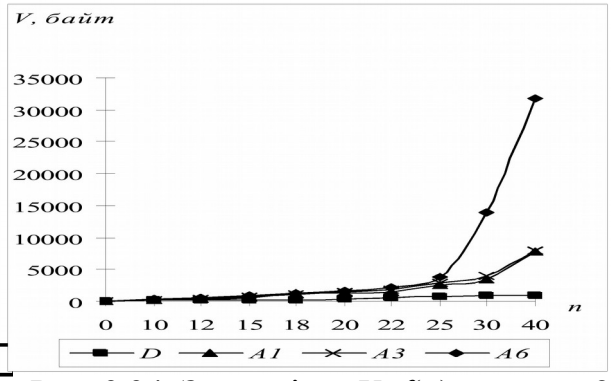


Рис. 3.34. Залежність $V=f(n)$ для зони 2

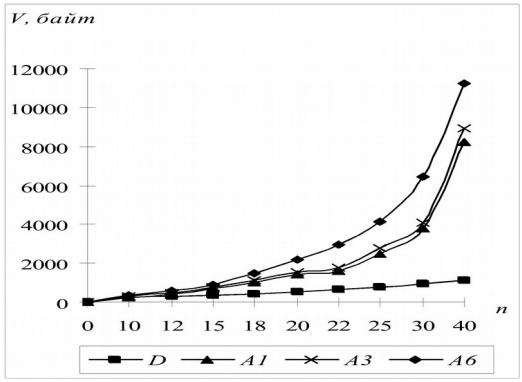


Рис. 3.35. Залежність $V=f(n)$ для зони 3

Часову складність алгоритмів A_1 і A_3 досить легко оцінити теоретично. Аналіз кожного вектора x включає одну операцію додавання й одну операцію порівняння, тобто дві елементарні операції. Максимальна кількість векторів у найгіршому разі буде на другому ранзі й становить $n^2/2$ (3.10). Кількість таких рангів у графі $D\Delta$ дорівнює n. Отже, часова складність алгоритму A_1 складе $O(2n \cdot n^2/2) = O(n^3)$, для зберігання векторів в Ω – областях на усіх рангах буде потрібно пам'яті не більше, ніж $O(n^3)$.

Оцінка часової складності алгоритму A_2 збігається з оцінкою для алгоритму A_1. Оцінимо часову складність алгоритму A_3. Часова складність алгоритму A_1 становить $O(n^3)$. Однак в A_3 кожна множина буде містити після виділення коридору й фільтрації в найгіршому разі два шляхи, що збільшить часову складність алгоритму A_3 у два рази й складе $O(2n^3) \cong O(n^3)$.

Однією з важливих характеристик наближених алгоритмів служить відносна похибка Δf, що він дає у випадку розв'язання задачі. З розглянутих алгоритмів наближеними є A_1 і A_3. Відносна похибка залежить від зони, у яку потрапить оптимальний розв'язок. Однак одержання неточних розв'язків алгоритмами A_1 і A_3 для різних зон у середньому викликало певні труднощі. Так, для зони 1 експонентне зростання векторів у графі $D\Delta$ натрапляє на обмеження (3.12) і тому в множинах оптимальні

шляхи будуть домінуючими по вагах функціонала. У зоні 3 оптимальний шлях буде проходити по більш високих вершинах графа DΔ, що не дозволить його відсікти. Тому на рис. 3.36 показана зміна похибки Δf і коефіцієнта $К_н$ залежно від розмірності n тільки для другої зони.

Як видно з рис. 3.37, для обох алгоритмів A_1 і A_3 середня похибка Δf стабілізується близько 5 % для першого алгоритму й близько 2 % для другого при n > 25.

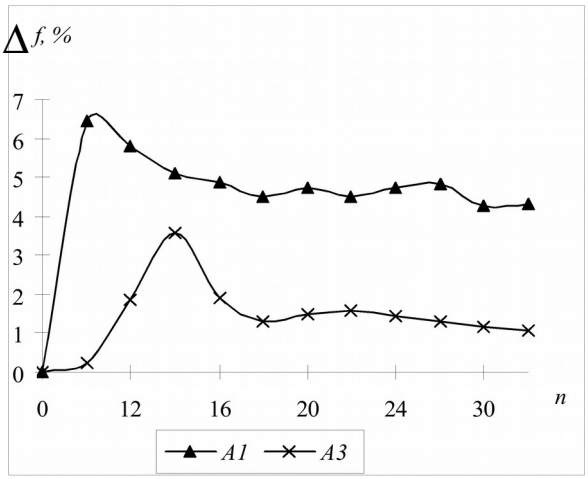


Рис. 3.36. Залежність Δf від n для зони 2

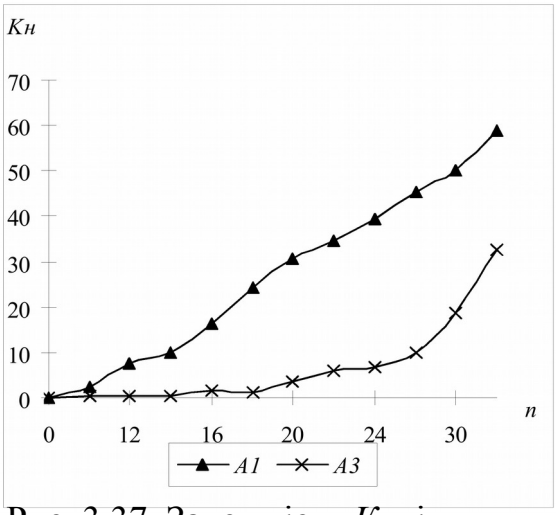


Рис. 3.37. Залежність $K_н$ від n для зони 2

Дослідження алгоритмів другої групи. Оцінимо властивості алгоритмів другої групи A_4, A_5, A_7, A_8 за обраними показниками ефективності. Алгоритми A_4 і A_5 є наближеними, а A_7 і A_8 – точними. Точно так, як і для одномірних задач, значення зазначених показників залежить від $\bar{r}$, що підтверджує графік на рис. 3.38.

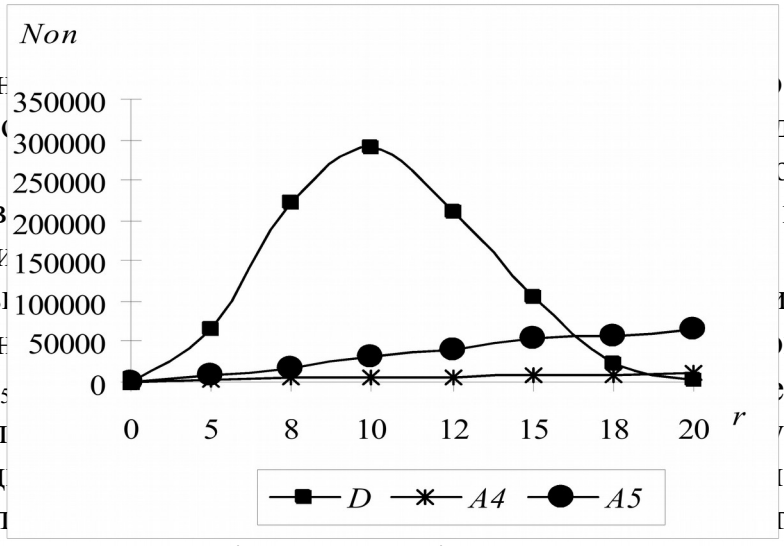


Рис. 3.38. Залежність N_{on} від для n=20 і m=5

Криві н… відповідним показникам … діл по зонах правочинний … ості обраних показників в… відповідними залежностями

У зв'язі… умови (3.8) часова склад… ова складність алгоритму A_5 …, ніж часова складність ал… що з кожної множини згід… ому разі на m векторів біл… грат пам'яті, використовуваної для зберігання даних, також в m раз збільшиться й дорівнює $O(mn^3)$. При m=5 зроблено дослідження похибки алгоритмів A_4 і A_5. Залежність f і $K_н$ від n і m для 2-ї зони зображено у вигляді графіків на рис. 3.39.

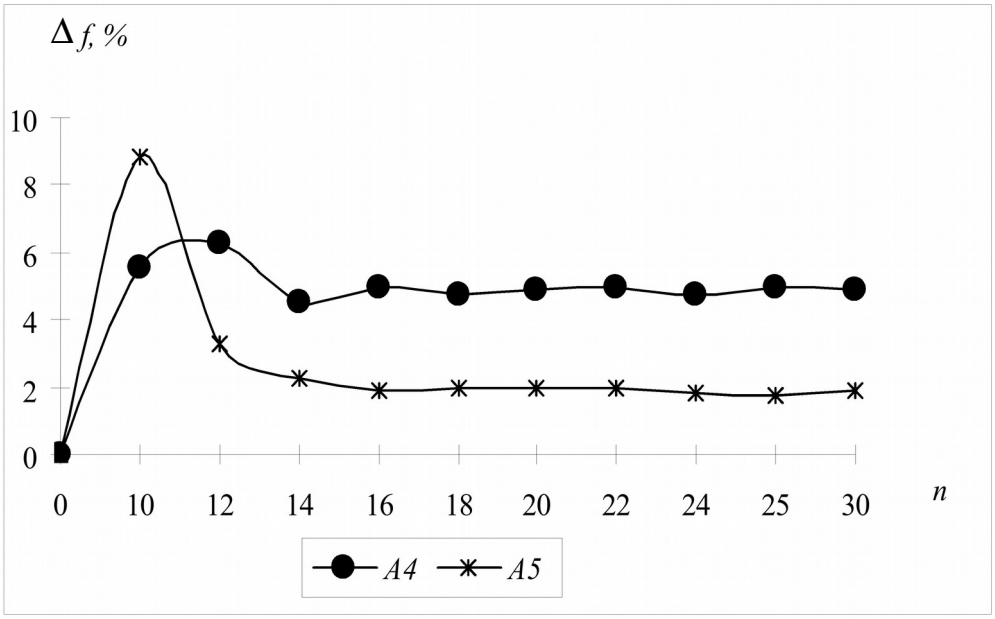
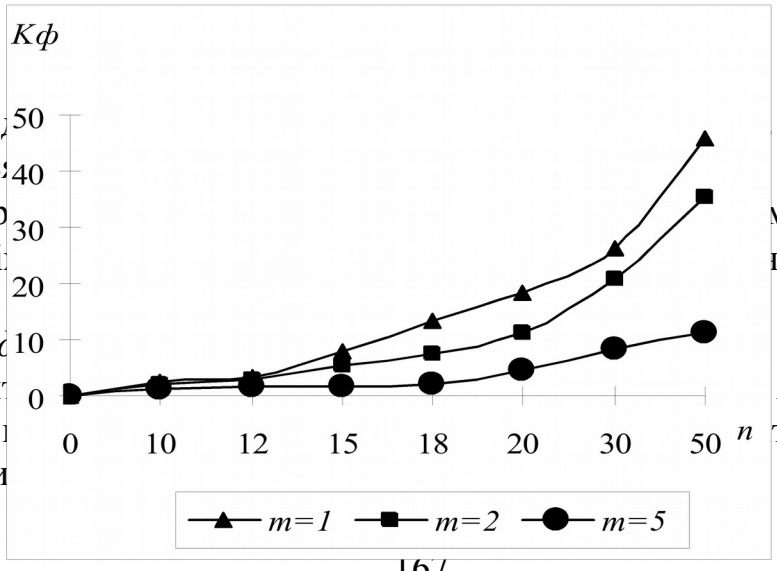


Рис. 3.39. Залежність Δf від n для зони 2

Якщо порівняти рис. 3.36 і 3.39, то видно, що відносна похибка Δf не залежить від m, а залежить тільки від обраного алгоритму. Таким чином, можна стверджувати, що алгоритм A_4 має похибку 5 %, а алгоритм A_5 – 2 %.

Для точних алгоритмів A_7 і A_8 використовується підхід з виділенням m-мірного коридору. При цьому отримані алгоритми мають експонентну складність, що полягає в експонентній кількості векторів, які можуть потрапити в цей коридор при побудові шляхів, починаючи з рангу $r = 1$. Тому використання цих алгоритмів більш ефективно при вже знайденому припустимому розв'язку яким-небудь із наближених алгоритмів, а оцінка значень показників ефективності буде зроблена при побудові багатоетапних точних алгоритмів.

Порівняємо алгоритми A_7 і A_8 між собою. Як основний критерій такого порівняння обрано коефіцієнт фільтрації $K_ф$. Для найгіршого розподілу (зона 2) залежність $K_ф$ від n показана на рис. 3.40.



Як ви прагне до 1. Це
пояснюється шляху в більш
високих вер менше значення
по вагах усі ала.

Дослід Проведемо
експеримент A_9, A_{10}, A_{11}, A_{12}.
Перші два тмів. Інші – до
багатомірни тестових задач

Рис. 3.40. Залежність $K_ф$ від n для зони 2

алгоритмом із заданою розмірністю вхідних параметрів n і m. Діапазон змін величин в обмеженнях підбирався так, щоб ранг оптимального розв'язку, одержуваного на останньому етапі алгоритму (він є точним), належав другій зоні, тобто $\bar{r} = \lfloor n/3 \div 2n/3 \rfloor$.

Результати дослідження показали (рис. 3.41, 3.42) істотний виграш алгоритмів A_9 і A_{10} порівняно з алгоритмом A_7. Тому використання значень отриманого припустимого розв'язку на попередніх етапах дозволяє зменшити кількість оброблюваних векторів алгоритмом. Аналогічні результати отримані й для алгоритмів A_{11} і A_{12}. Швидкість розв'язання на останньому етапі для алгоритмів A_{11} і A_{12} залежить від максимальної кількості векторів, оброблюваних на одному ранзі $\bar{r}$.

Такий самий експеримент був проведений для алгоритмів A_{11} і A_{12} (рис. 3.43, 3.44). Тільки тепер змінювався вхідний параметр m, що задає кількість обмежень. Результати цього експерименту для найгіршого випадку (зони 2) зображені на рис. 3.43 та 3.44 відповідно.

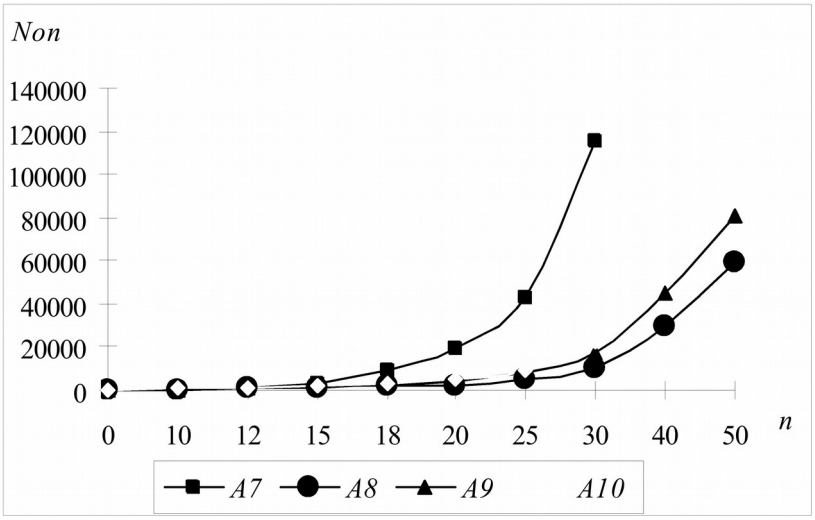


Рис. 3.41. Залежність N_{on} від n для зони 2

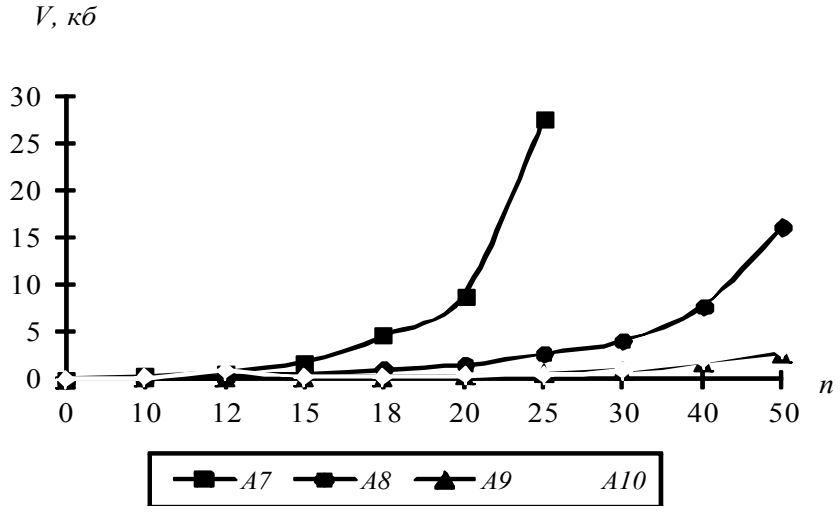


Рис. 3.42. Залежність V від n для зони 2

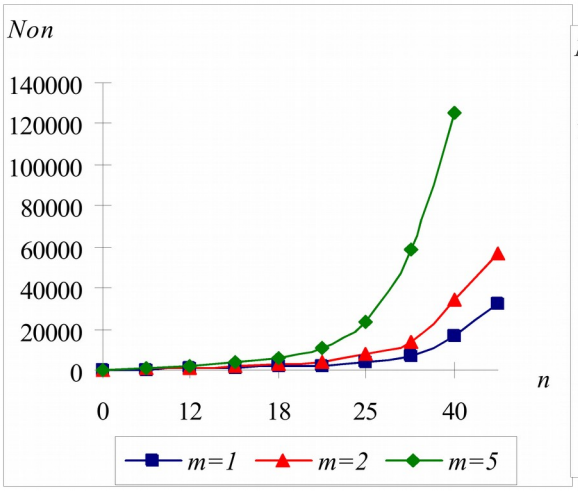


Рис. 3.43. Залежність N_{on} від n та m для алгоритму A_{11}

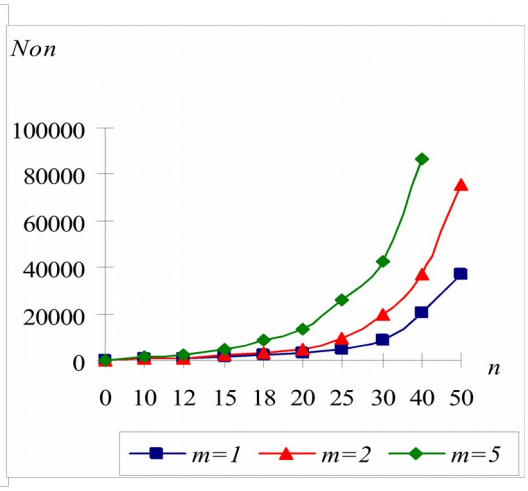


Рис. 3.44. Залежність N_{on} від n та m для алгоритму A_{12}

Таким чином, у цьому пункті експериментально досліджено властивості розроблених алгоритмів і дано оцінку показників ефективності для них. З метою визначення практичної значущості алгоритмів A_1 – A_{12} варто зробити їхній порівняльний аналіз за обраними показниками з відомими.

3.6.3. Порівняльний аналіз рангових алгоритмів з відомими алгоритмами

Порівняльний аналіз зробимо у два етапи, що відповідають порівнянню одномірних і m–мірних алгоритмів з відомими.

Етап 1. Розробленими одномірними алгоритмами в роботі є алгоритми A_1, A_2, A_3, що дають наближений розвязок, і A_6, A_9, що дають точний розв'язок. З відомих точних одномірних алгоритмів виберемо адитивний алгоритм Балаша [75], що має експонентну складність, і алгоритм ДП [187], що має часову складність $O(cn^2)$, де c – абсолютне значення цільової функції. Як показав експеримент (рис. 3.45), алгоритм ДП також залежить від $\bar{r}$.

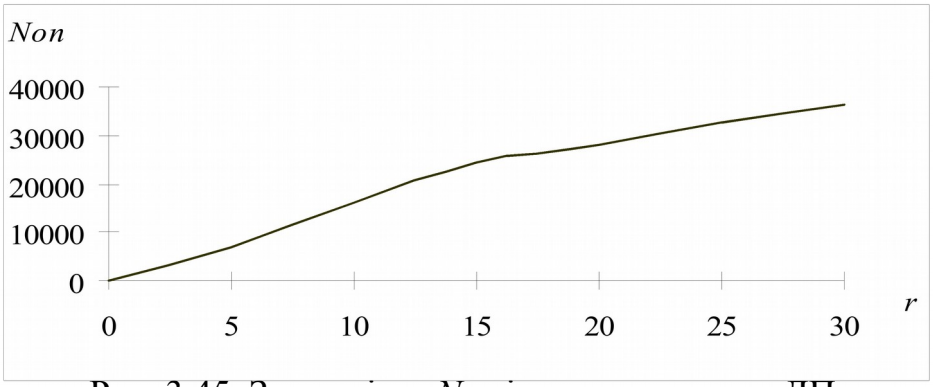


Рис. 3.45. Залежність N_{on} від для алгоритму ДП

Таким чином, можна зробити висновок: часова складність точних алгоритмів для розв'язання задачі ЦЛП із БЗ залежить від приналежності оптимального розв'язку до якої-небудь зони. Звідси випливає, що для точних алгоритмів розв'язання задач ЦЛП із БЗ існує хоча б одна зона, у якій спостерігається експонентне зростання виконуваних ЕО, що залежать від кількості оброблюваних припустимих розв'язків. Приналежність до виділеної зони визначається кількістю одиниць в оптимальному розв'язку.

У випадку порівняння розроблених точних одномірних алгоритмів A_6, A_9 і A_{10} з алгоритмом Балаша й алгоритмом динамічного програмування можна зробити такі висновки.

При потраплянні оптимального розв'язку в першу зону для задач невеликої розмірності (n<30) з погляду часу розв'язання найбільш ефективними є алгоритми Балаша й ДП (для цього алгоритму додається умова малого абсолютного значення функціонала). У випадку, коли n>30 і оптимальний розв'язок потрапляє в першу зону, ефективніше застосовувати багатоетапний алгоритм A_{10}. Якщо оптимальний розв'язок належить другій і третій зоні, то переважнішими є багатоетапні алгоритми A_9 і A_{10}, причому за часовими показниками спостерігається експонентне зростання виграшу цих алгоритмів над алгоритмом Балаша залежно від розмірності задачі n. Застосування алгоритму A_6 неефективно, тому що без початкового припустимого розвязку йому доводиться перебирати експонентну кількість векторів, що лежать усередині коридору. Використання стратегій L_6, L_8, L_9 призводить до виділення коридору усередині множин, а таке виділення є менш ефективним.

Так, для точного розв'язку одномірних задач маленької розмірності й оптимального розв'язку, що лежить усередині першої зони, виявляється переважніше алгоритм Балаша. В інших випадках доцільніше застосовувати алгоритми A_9 і A_{10}.

Якщо необхідно швидко знайти припустимий розв'язок, то за поліноміальний час алгоритмом A_1 буде отримано такий розв'язок з похибкою близько 5 %, а алгоритмом A_3 - з похибкою близько 2 %, на що буде потрібно у 2 рази більше часу.

Eman 2. Перейдемо до розгляду алгоритмів другої групи. Із числа розроблених алгоритмів до них належать наближені (A_4, A_5) і точні (A_7, A_8, A_{11}, A_{12}). Як видно з рис. 3.40, застосування стратегії виділення m–мірного коридору на всьому ярусі переважніше, ніж його виділення в множинах. Однак застосовувана стратегія відсікань L_{13} в алгоритмі A_8 використовується й у багатоетапних алгоритмах A_{11} і A_{12} для одержання точного розв'язку. Тому для порівняння з відомими алгоритмами розв'язання задачі ЦЛП із БЗ будуть обрані алгоритми A_{11} і A_{12}.

Почнемо порівняльний аналіз із тих методів і алгоритмів, які дають точний розв'язок задачі (3.7) – (3.9). До таких методів належать адитивний алгоритм Балаша [75], його модифікація для задач із додатними

коефіцієнтами [113], метод *W* послідовного аналізу й відсікання варіантів [88] і *P*-метод. У табл. 3.1 наводяться часові значення розв'язання задачі (3.7)–(3.9) зазначеними методами і їхній виграш відносно алгоритму Балаша. За даними табл. 3.1 можна зробити такий висновок: *W*-метод поступається алгоритму Балаша на малих і середніх розмірностях і більш ефективний для задач великих (n>100) розмірностей. У той же час на малих і середніх розмірностях більш ефективним є *P*-метод, а при збільшенні розмірності він втрачає свою перевагу. Отже, якщо показати виграш розроблених алгоритмів A_{11} і A_{12} відносно алгоритму Балаша й указати значення цього виграшу, то за аналогією можна зробити порівняння й з алгоритмами *W*–методу і *P*-методу.

Таблиця 3.1

Часові оцінки відомих алгоритмів

n	m	Загальний час рахунку, с			Відношення до алгоритму Балаша	
		алгоритм Балаша	алгоритм методу W	алгоритм P–методу	алгоритм методу W	алгоритм P–методу
10	20	38	100	12	2.63	0.31
10	50	105	202	23	1.92	0.21
10	100	241	294	23	1.21	0.09
50	10	42	42	17	1	0.4
50	50	190	448	182	2.35	0.95
100	10	62	55	32	0.88	0.52

Найбільший виграш $K_в$ в алгоритму A_{12} над алгоритмом Балаша спостерігається в другій зоні (рис. 3.29 та 3.44), коли кількість оброблюваних векторів максимальна. У першій зоні час знаходження оптимального розв'язку порівняно з часом розв'язання для алгоритму Балаша. Якщо оптимальний розв'язок належить третій зоні, то саме тут проявляються переваги алгоритму Балаша ($K_в$<1).

Таким чином, можна зробити такі висновки.

1. Для задач маленької розмірності й при потраплянні точного розв'язку в першу зону ефективніше використовувати P–метод.

2. Для задач великої розмірності й при потраплянні в першу зону – алгоритми A_{11} і A_{12}. При потраплянні оптимального розв'язку в другу зону найбільшого часового виграшу при незначних витратах пам'яті мають алгоритми A_{11} і A_{12}.

3. Для задач маленької розмірності (n < 50) і при потраплянні оптимального розв'язку в третю зону переважнішим є алгоритм Балаша, що зі зростанням n втрачає його. Тому для задач великої розмірності й розв'язків, що належать третій зоні, доцільніше застосовувати алгоритми, засновані на ранговому підході.

3.6.4. Аналіз впливу сортувань коефіцієнтів при функціоналі й обмеженнях на величину похибки розв'язання

Покажемо, що алгоритмам A_1, A_2, A_3 властивою є стабілізація похибки зі збільшенням розмірності задачі (рис. 3.36). Становить інтерес також експериментальне дослідження впливу на похибку наближених алгоритмів A_1, A_2, A_3 різних видів сортувань коефіцієнтів при функціоналі й обмеженні (3.5):

$$a_1 \le a_2 \le a_3 \le \ldots \le a_n ; \tag{3.78}$$

$$c_1 \ge c_2 \ge c_3 \ge \ldots \ge c_n ; \tag{3.79}$$

$$\frac{c_{11}}{a_{11}} \ge \frac{c_{12}}{a_{12}} \ge \frac{c_{13}}{a_{13}} \ge \ldots \ge \frac{c_{1n}}{a_{1n}} . \tag{3.80}$$

Для одержання точного розв'язку як еталон використовуємо алгоритм A_7. Всі результати отримано з довірчою ймовірністю 0,95. Коефіцієнти у функціоналі й обмеженнях генеровані за рівномірним законом розподілу в діапазоні (1–100). АпроКЗимуючі криві результатів обчислень похибок і кількості неточних розв'язків для алгоритмів A_1, A_2, A_3 подані на рис. 3.46 – 3.48 відповідно.

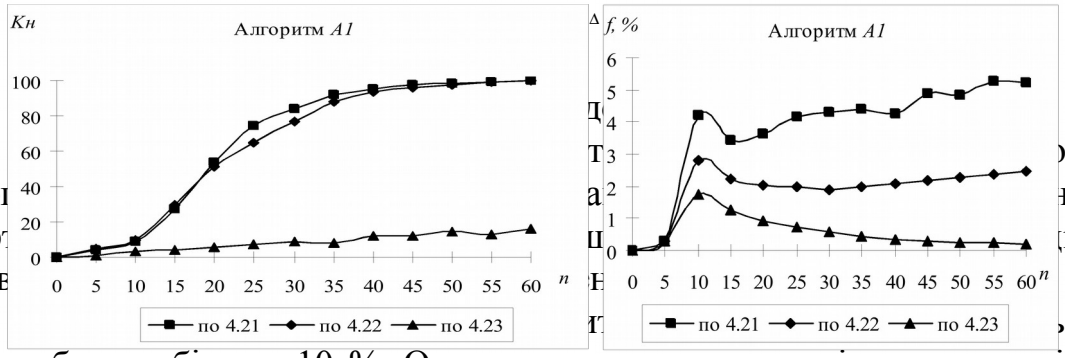


Рис. 3.46. Залежність $K_н$ та Δf для алгоритму A_1

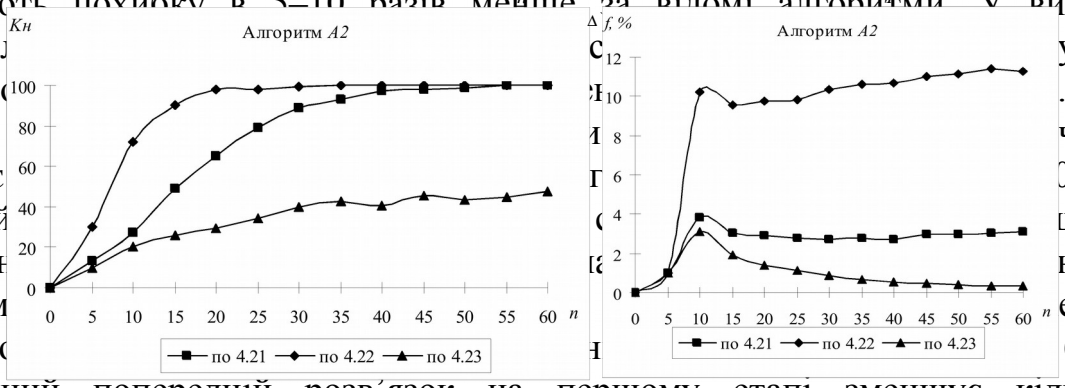


Рис. 3.47. Залежність $K_н$ та Δf для алгоритму A_2

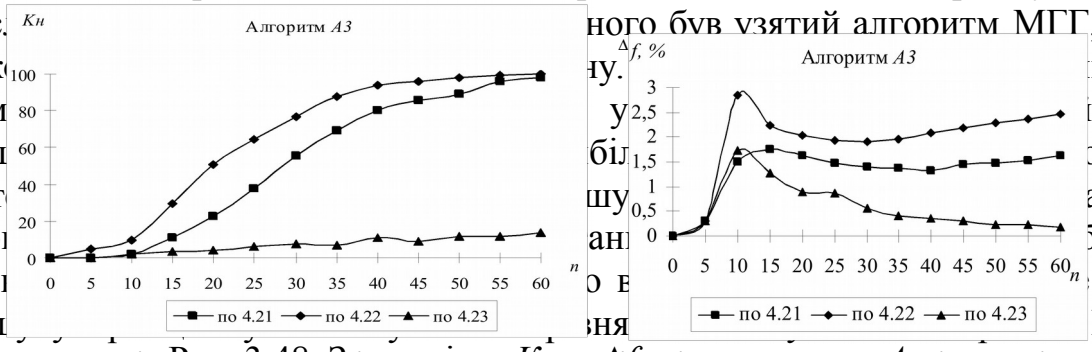


Рис. 3.48. Залежність $K_н$ та Δf для алгоритму A_3

малих розмірностях це дійсно так. Але зі збільшенням розмірності задачі об'єм проміжних даних зростає настільки, що ОЗП для їхнього зберігання виявляється недостатньо. Тому ці дані доводиться зберігати на ЗЗП, що

істотно збільшує час розв'язання задачі, тому що операція читання й запису на ЗЗП істотно більше, ніж в ОЗП. Крім того, різні модифікації МГГ, що визначають до початку роботи основного алгоритму верхню оцінку розв'язку, дозволили практично вирівняти кількість аналізованих варіантів для обох стратегій. Оскільки ж нас у першу чергу цікавили задачі великої розмірності, то в якості еталонного був обраний МГГ зі стратегією пошуку вглибину. Використання МГГ дозволяє зменшити кількість аналізованих варіантів, але проблема полягає в тому, що, навіть одержавши розв'язок, алгоритм продовжує перевіряти ті гілки дерева шляхів, вага (ранг) яких менше від отриманого розв'язку. Таким чином, навіть якщо припустити, що верхня оцінка розв'язку збіглася з оптимальним значенням R^*, кількість векторів (вузлів розгалуження) $\sum_{r=1}^{R^*} C_n^r$, побудованих МГГ, буде дорівнювати сумі R^* елементів ряду трикутника Паскаля.

Алгоритми ж, засновані на ранговому підході, за рахунок попереднього калібрування й прогнозування кінцевого результату дозволяють на кожному ранзі, починаючи з першого, відсікати ті шляхи, які алгоритм визначає як безперспективні.

На рис. 3.49 показана залежність кількості оброблених векторів від рангу розв'язків задачі розмірності 20x200 для описаних алгоритмів. Для наочності на цьому ж графіку наведена крива "трикутника Паскаля".

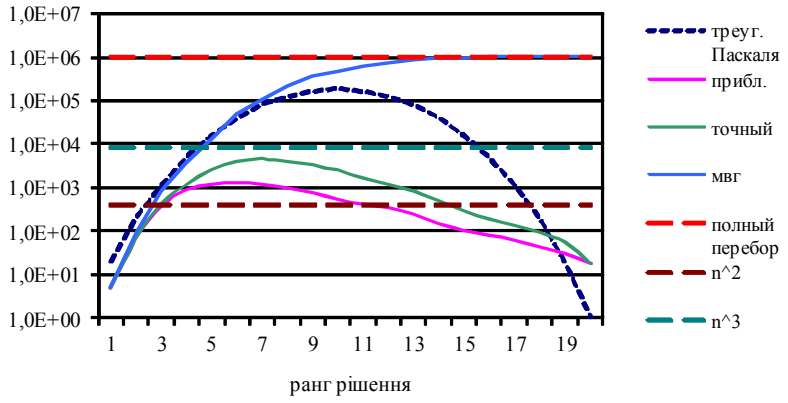


Рис. 3.49. Залежність кількості оброблених векторів від рангу рішення задачі

Під безперспективними шляхами розуміються шляхи, які в принципі не можуть дати покриття, а також ті, які дадуть покриття на ранзі, що перевищує прогнозований оптимальний розв'язок. Інакше кажучи, на відміну від МГГ, відсікаються не тільки ті гілки, довжина яких більше оптимального значення, але й ті, які не містять розв'язків у принципі.

Криві на графіках підтверджують, що складність розглянутих наближених алгоритмів не може перевищувати $O(n^3m)$, оскільки всього ярусів n, на ярусі кількість сформованих векторів не перевищує n^2, і

кількість елементів, аналізованих у кожному векторі, не перевищує m. Для точних алгоритмів отримана оцінка часової складності в середньому також відповідає $O(n^3 m)$. Що ж стосується МГГ, то зі збільшенням рангу рішення МГГ наближається до повнопереборного алгоритму. Для справедливості слід зазначити, що кількість елементарних операцій, необхідних для побудови одного вектора в МГГ менше, ніж в алгоритмах, заснованих на ранговому підході. Це компенсується зменшенням кількості оброблюваних векторів. Графіки залежностей кількості елементарних операцій і часу розв'язання від рангу шляху подані на рис. 3.50 і 3.51.

Що ж стосується залежності похибки наближеного алгоритму від рангу шляху, то вона виникає при потраплянні розв'язку в другу умовно виділену зону й коливається в межах від 0 до 15 % (рис. 3.52), причому кількість неточних розв'язків не перевершує 10 % загальної кількості дослідів.

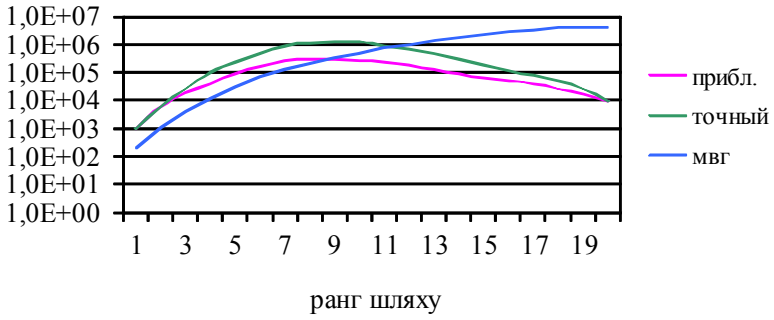


Рис. 3.50. Залежність кількості елементарних операцій від рангу шляху

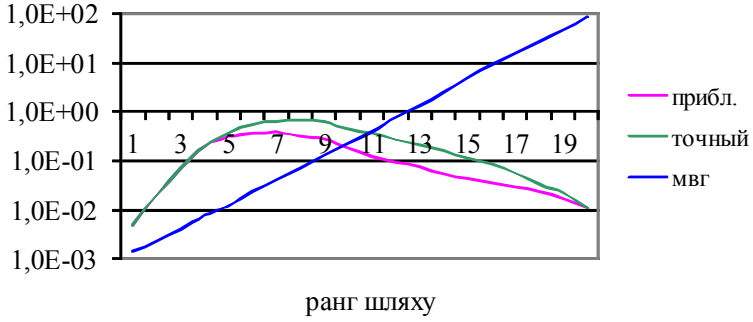


Рис. 3.51. Залежність часу розв'язання від рангу шляху

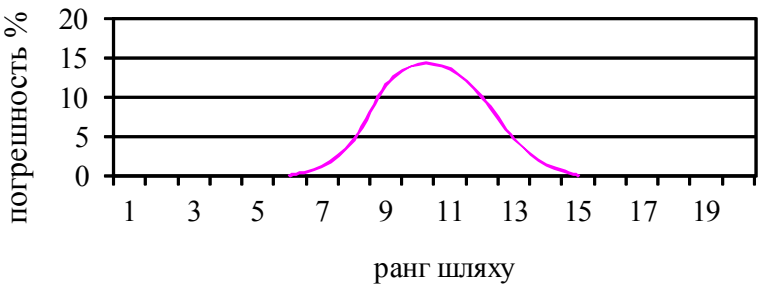


Рис. 3.52. Залежність похибки від рангу шляху

Для оцінки залежності кількості елементарних операцій, кількості оброблених векторів і часу розв'язання від розмірності задачі вихідні дані за рахунок щільності заповнення одиницями генерувалися таким чином, щоб розв'язок потрапляє в першу й другу умовно виділені зони, тому що для третьої зони перевага розроблених у роботі алгоритмів очевидна. З рис. 3.53–3.55 видно, що зі збільшенням розмірності задачі підхід до розв'язання ЗНП на основі методу галузей і границь із погляду часової і алгоритмічної складності істотно поступається методам, заснованим на ранговому підході.

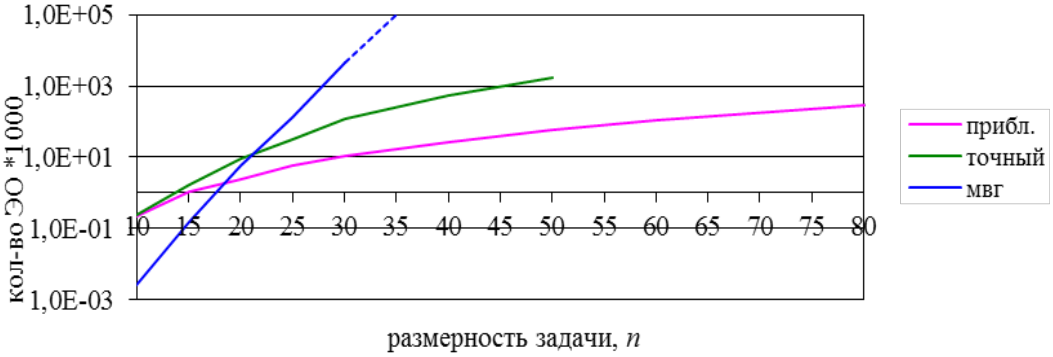


Рис. 3.53. Залежність кількості ЕО від розмірності задачі

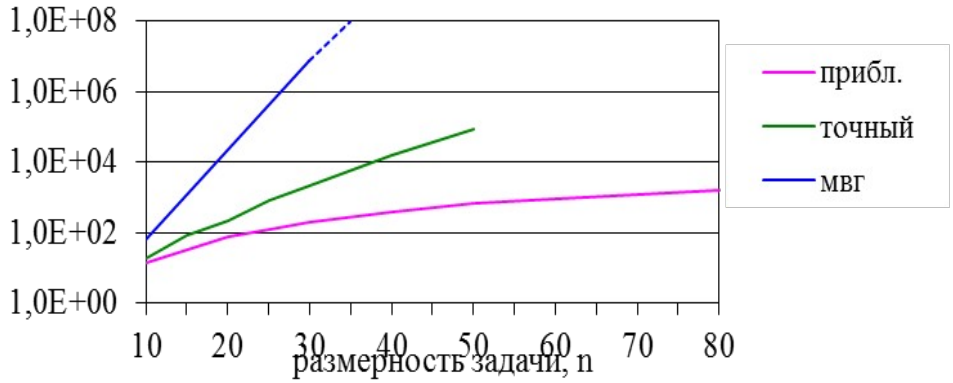


Рис. 3.54. Залежність кількості оброблених векторів від розмірності задачі

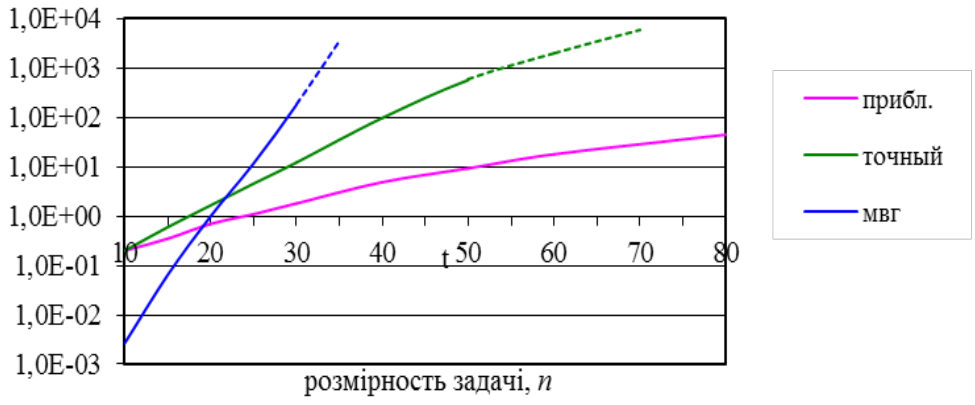


Рис. 3.55. Залежність часу розв’язання від розмірності задачі

Похибка наближених алгоритмів (рис. 3.56 – 3.58) стабілізується зі збільшенням розмірності розв'язуваної задачі. Так, при $n > 30$ вона не перевищує 6 %, а починаючи з $n = 50$ похибка не перевищує 3 %, тобто зі збільшенням n спостерігається збіжність наближеного розв’язку до оптимального, однак кількість неточних розв’язків зросла до 30 %.

З рис. 3.57 та 3.58 видно, що використання сортувань векторів із множини B у порядку зменшення відношення ваги стовпця до кількості одиниць у стовпці дозволять в 1,4 разу зменшити похибку розв’язання, яка починаючи з n=45 не перевищує 2 %, причому кількість неточних розв’язків не перевищує 10 % загальної кількості дослідів. На рис. 3.59 - 3.61 наведено аналогічні характеристики й для ЗНР.

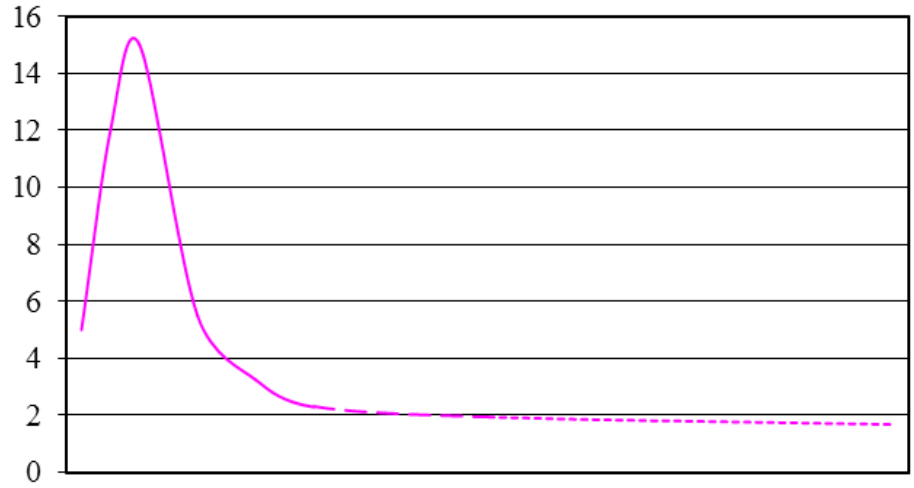


Рис. 3.56. Залежність похибки наближених алгоритмів від розмірності задачі

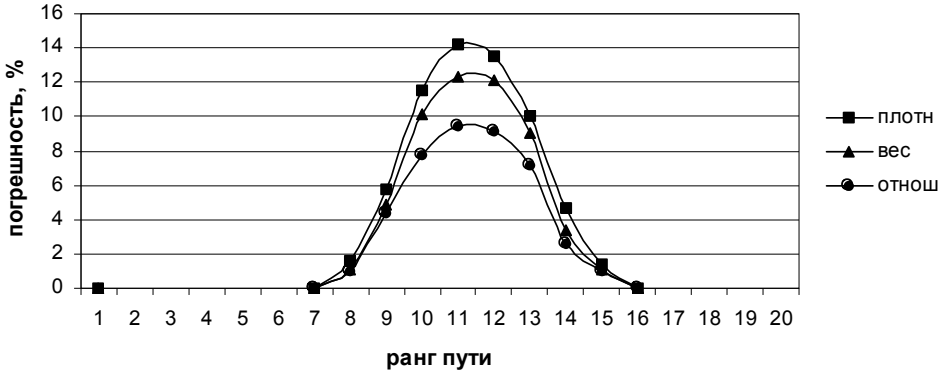


Рис. 3.57. Залежність похибки розв’язання ЗНП від рангу шляху для різних сортувань

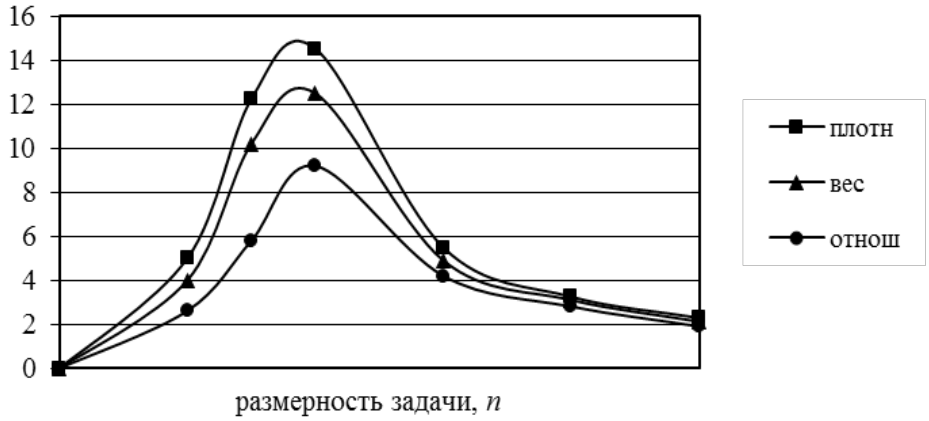


Рис. 3.58. Залежність похибки розв'язання ЗНП від розмірності задачі для різних сортувань

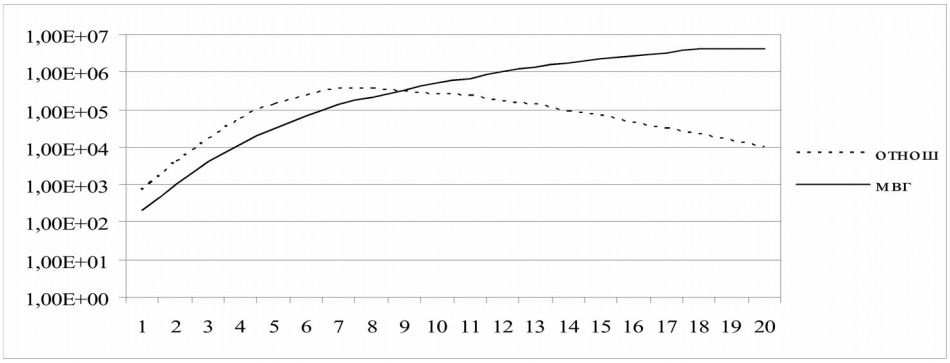


Рис. 3.59. Залежність кількості елементарних операцій від рангу шляху для ЗНР

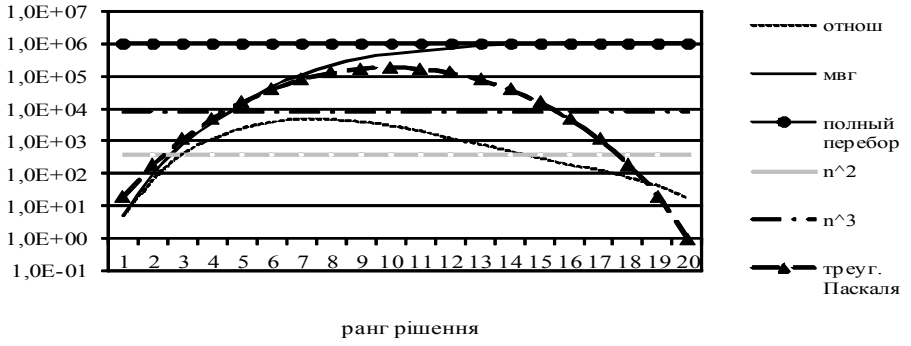


Рис. 3.60. Залежність кількості оброблених векторів від рангу шляху для ЗНР

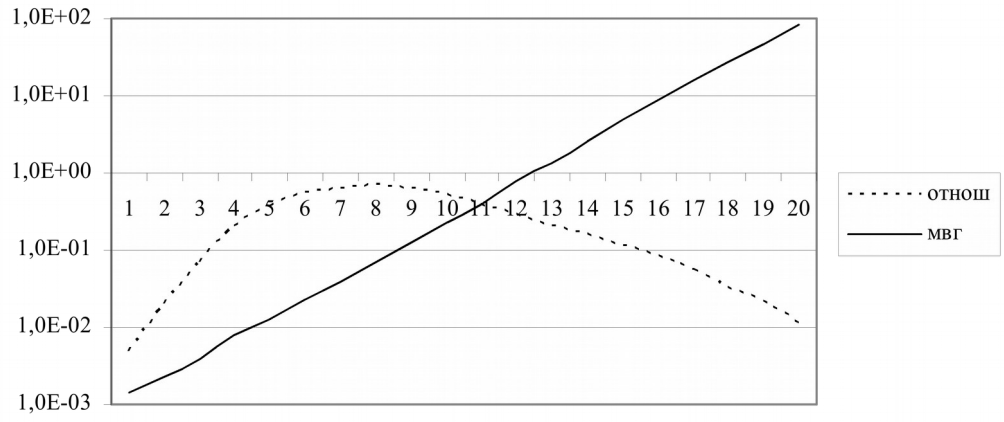


Рис. 3.61. Залежність часу розв'язання від рангу шляху для ЗНР

Дослідження залежності кількості операцій додавання й порівняння від кількості змінних у рівнянні при розв'язанні діофантових рівнянь алгоритмом виділення коридору, що реалізує ідею, показало, що при n =150 часова складність алгоритму не перевищувала $O(n^5)$, а при n не перевищувала $35–O(5999n^4+49)$. Залежність МО й СКО кількості елементарних операцій від n наведені на рис. 3.62 – 3.64.

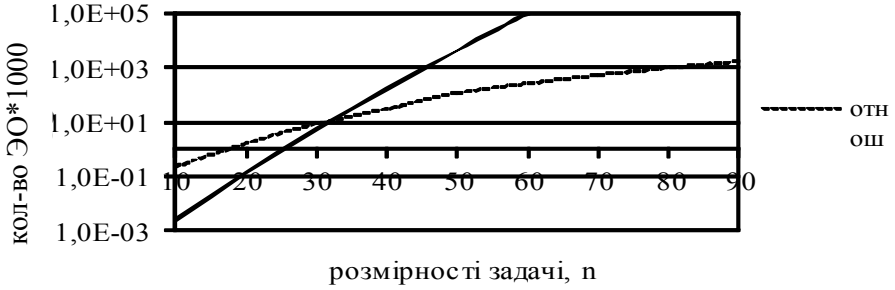


Рис. 3.62. Залежність кількості ЕО від розмірності задачі ЗНР

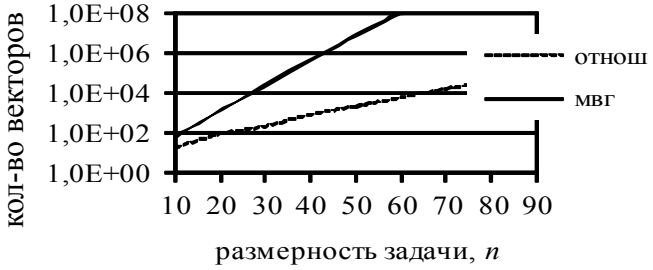


Рис. 3.63. Залежність кількість оброблених векторів від розмірності задачі для ЗНР

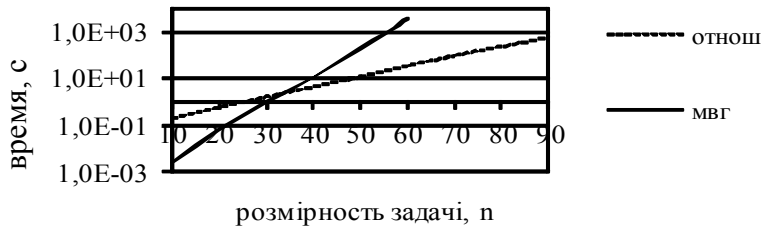


Рис. 3.64. Залежність часу розв'язання від розмірності ЗНР

3.6.5. Гарантовані прогнози і їхній вплив на часову складність і похибку рангових алгоритмів розв'язання задачі 0,1-рюкзак

В основі математичної моделі рангового підходу для побудови алгоритмів розв'язання задач ЦЛП із БЗ покладено принцип оптимізації по напрямку в дискретному просторі станів, заданому графом G_Δ [106]. Основна стратегія відсікання безперспективних варіантів розв'язань використовувана в алгоритмах, розроблених у роботі [106], заснована на перевірці нерівності

$$d_c(\mu_{sj}^r) + \gamma_p < \max_{\{c_j\}}\{d_c(\mu_{sp}^{*r})\},$$

де — $d_c(\mu_{sp}{}^r)$ довжина шляху $\mu_{sp}{}^r$ до вершини p рангу r по вагах $\{c_i\}$, що дозволяє виключити цей шлях з подальшого аналізу як безперспективний, якщо умова виконана;

$\gamma_p = c_{p+1} + c_{p+2} + \ldots + c_n$, $\gamma_n = 0$; $\delta = \overline{(1, n-1)}$ – представляє верхню оцінку (негарантований прогноз) приросту величини значення локального екстремуму в області Ω_p на усіх наступних рангах. У даному посібнику показано, що заміна використовуваного в алгоритмах [102–104] негарантованого оптимістичного прогнозу збільшення довжини шляхи в графі G_Δ на гарантований песимістичний дозволяє істотно знизити похибку наближених алгоритмів і часову складність точних алгоритмів, а також досліджено вплив різних сортувань коефіцієнтів у функціоналі й обмеженнях на похибку наближених алгоритмів.

Формалізація й розв'язання задачі. Для формування гарантованого прогнозу введемо матрицю $П$ з n-стовпцями й p-рядками ($p = m+1$) і вектор стовпець b_j правих частин обмежень (3.2):

$$П = \begin{array}{c|cccc|c|c} 1 & C_1 & C_2 & ..... & C_n & & b_j \\ 2 & a_{11} & a_{12} & ..... & a_{1n} & & b_1 \\ 3 & a_{21} & a_{22} & ..... & a_{2n} & ; & b_2 \\ ... & ..... & ..... & ..... & ..... & & .... \\ p & a_{m1} & a_{m2} & ..... & a_{mn} & & b_m \end{array}, \qquad (3.81)$$

де C_i– коефіцієнти при змінних x_i у функціоналі (3.1);

a_{ij} – коефіцієнти при змінних x_i в j–му обмеженні.

На основі матриці (3.81) побудуємо процедуру *A*, на основі якої будемо визначати гарантований прогноз (рис. 3.65).

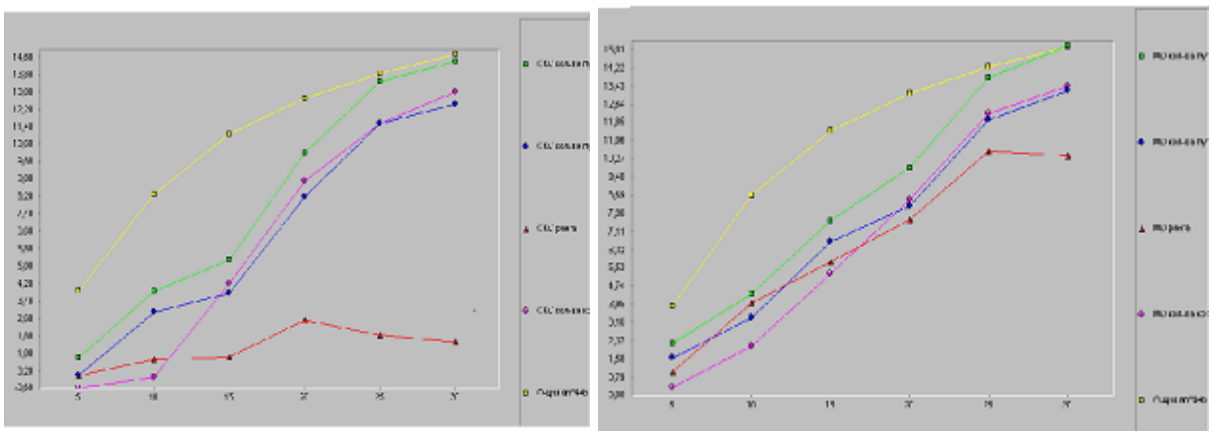


Рис. 3.65. Залежність МО й СКО кількості елементарних операцій при розв'язанні діофантових рівнянь із БЗ від кількості змінних у рівнянні

Процедура А

Крок 1. Змінній Z привласнимо значення, що дорівнює 2 в рядку $h= Z$ і відсортуємо стовпці матриці П у порядку зростання елементів a_{ij} у рядку *h*. Формуємо порожні множини *R* і *S*.

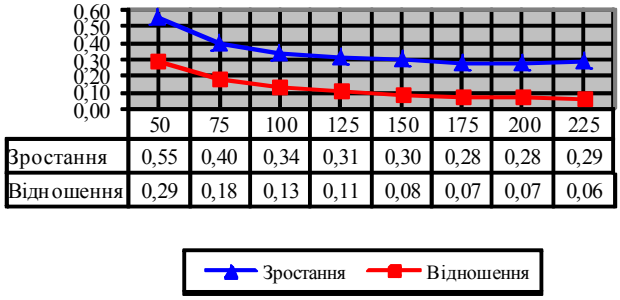
Крок 2. Здійснюємо підсумовування елементів стовпців, починаючи з перших двох стовпців у відсортованій матриці П і перевіряючи для кожного рядка виконання нерівності $\sum_{i=1}^{n} a_{ij} \le b_j \quad j = \overline{(1,m)}$. Підсумовування стовпців припиняємо при досягненні n значення r–1, починаючи з якого для одного з рядків j нерівність $\sum_{i=1}^{n=r-1} a_{ij} \le b_j$ перестане виконуватися й значення r заносимо на вільну позицію в множину *R* (r–кількість додавань стовпців до моменту, коли в одному з рядків нерівність $\sum_{i=1}^{n} a_{ij} \le b_j, j = \overline{(1,m)}$ не виконується), а суму відповідних c_i, накопичену по першому рядку, розміщаємо на вільну позицію в множині *S*.

Крок 3. Перевіряємо $Z = p$, якщо так, то переходимо до виконання наступного кроку, інакше привласнюємо $Z = Z + 1$ і переходимо до виконання кроку 2.

Крок 4. У множині *R* знаходимо максимальний елемент r_{max}, розташований на *к*-й позиції, і суму s_k, розташовану на *к*-й позиції в множині *S*.

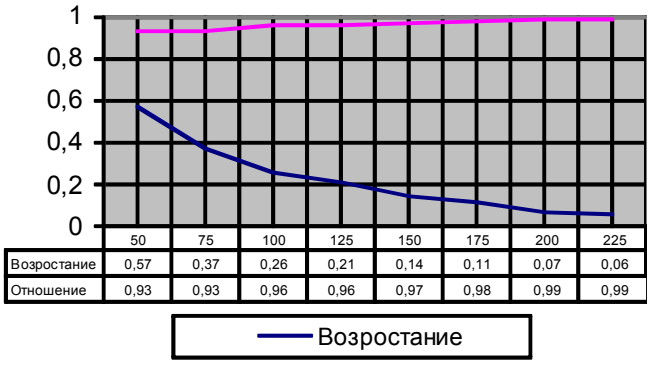
Отримана пара значень (r_{max}, s_k) є гарантованим прогнозом того, на скільки рангів у графі G_Δ можна продовжити шлях, не порушуючи обмежень (максимальна кількість вершин, яку можна ще приєднати до шляху, не порушуючи обмежень), при цьому поточна сума s_k на κ-й позиції множини S дає нижню гарантовану оцінку довжини шляху по вагах функціонала. У процедурі A на кроці 4 можливий також вибір максимальної суми s^*_k і відповідного їй значення r_κ, при цьому пари (s^*_k, r_κ) є теж гарантованим прогнозом того, що до шляху можна приєднати r_κ вершин, не порушуючи обмежень, і при цьому максимальна нижня гарантована оцінка довжини шляхи по вагах функціонала дорівнює s^*_k. Причому якщо в графі G_Δ процедура A застосовується для конкретної вершини і, то стовпці з номерами меншими, ніж і, виключаються з матриці П, оскільки, як видно з графа G_Δ (рис. 3.4), вершини з номерами меншими і вже не можуть зустрічатися в шляху. У випадку одномірної задачі матриця П буде містити тільки два рядки. В алгоритмах розв'язання задачі (3.7)–(3.9), наведених у роботі [106], для підвищення ефективності відсікання безперспективних варіантів розв'язків уведено калібровані вектори, за якими здійснювалася оцінка того, скільки шлях може набрати по функціоналі, при цьому вони теж ґрунтуються на негарантованому оптимістичному прогнозі γ_p для довільної вершини р. Застосування процедури А дозволяє для кожної вершини графа G_Δ створити калібровані вектори шляхом сортування стовпців матриці П у порядку зростання a_{ij} для одномірних задач і в порядку зменшення c_j для багатомірних задач. У роботах [101–105] показано, що при розв'язанні одномірних задач найбільш ефективними сортуваннями коефіцієнтів функціонала є сортування в порядку зменшення ваг функціонала й у порядку зменшення відношення коефіцієнтів при x_i у функціоналі до відповідних коефіцієнтів при x_i в обмеженні. Причому в останньому випадку похибка була найменшою, а у випадках застосування стратегій *max* і *max–min* не перевищувала 3–5%. Тому цікаво оцінити, як вплине заміна негарантованого оптимістичного прогнозу на гарантований песимістичний прогноз і зміна сортувань у каліброваних векторах на основі запропонованої процедури А на часову складність і похибку алгоритмів, розроблених у роботі [101–105]. Порівняння наближених алгоритмів проводилося з алгоритмом, розробленим у роботі [101], що реалізує стратегію ***max,*** а в якості точного алгоритму, для порівняння, використовувався двохетапний алгоритм, наведений у роботі [106]. Експериментально отримано оцінку в середньому для часової складності аналізованих алгоритмів і їхньої похибки. Оцінка в середньому здійснювалася шляхом розв'язання 1000 тестових задач для кожної точки графіка, при цьому ваги коефіцієнтів у функціоналі й обмеженнях генерувалися за рівномірним законом розподілу. Середні значення часової складності й відносної похибки алгоритмів отримані з довірчою

ймовірністю 0,95. Результати експериментального дослідження подано на рис. 3.66–3.73.



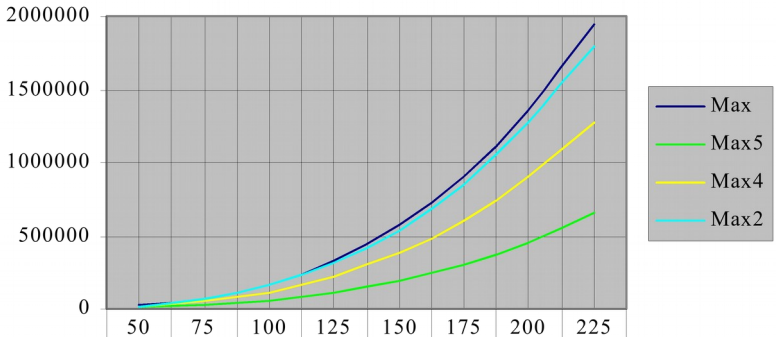
	50	75	100	125	150	175	200	225
Зростання	0,55	0,40	0,34	0,31	0,30	0,28	0,28	0,29
Відношення	0,29	0,18	0,13	0,11	0,08	0,07	0,07	0,06

Рис. 3.66. Залежність похибки алгоритму Max2 від початкового сортування функціонала



	50	75	100	125	150	175	200	225
Возростание	0,57	0,37	0,26	0,21	0,14	0,11	0,07	0,06
Отношение	0,93	0,93	0,96	0,96	0,97	0,98	0,99	0,99

Рис. 3.67. Залежність частоти збігу розв’язання наближеного алгоритму Max2 з точним розв’язком від способу сортування функціонала



	50	75	100	125	150	175	200	225
Max	21134	71426	2E+05	3E+05	6E+05	9E+05	1E+06	2E+06
Max5	7834	25200	58551	1E+05	2E+05	3E+05	4E+05	7E+05
Max4	15182	49825	1E+05	2E+05	4E+05	6E+05	9E+05	1E+06
Max2	20189	67734	2E+05	3E+05	5E+05	9E+05	1E+06	2E+06

Рис. 3.68. Середня кількість оброблюваних векторів для наближених алгоритмів

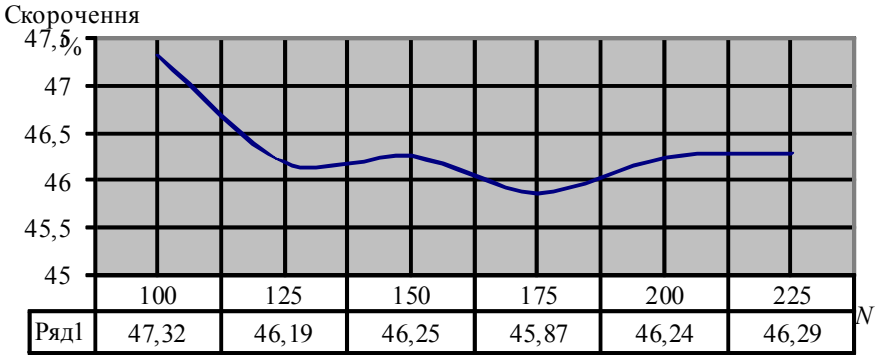


Рис. 3.69. Залежність скорочення кількості аналізованих векторів при заміні негарантованого прогнозу на гарантований прогноз у точному методі

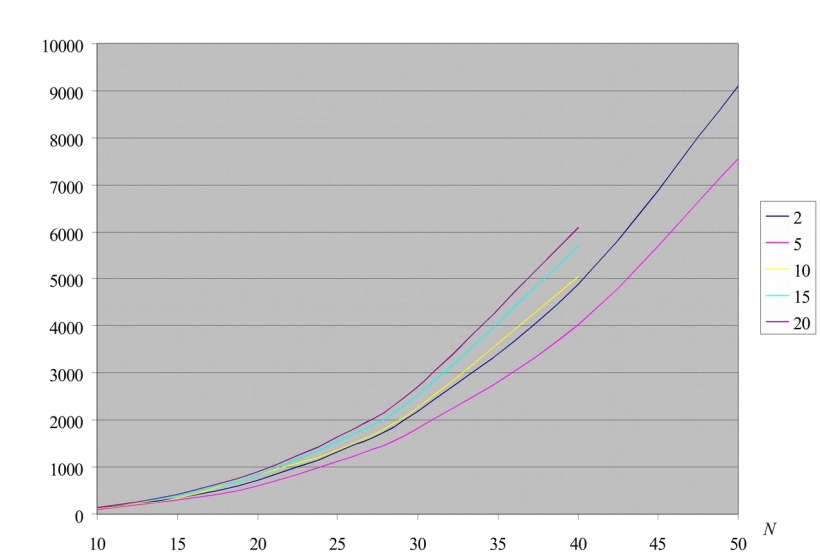


Рис. 3.70. Залежність середньої кількості аналізованих векторів від розмірності задачі при різній кількості обмежень для MAX5

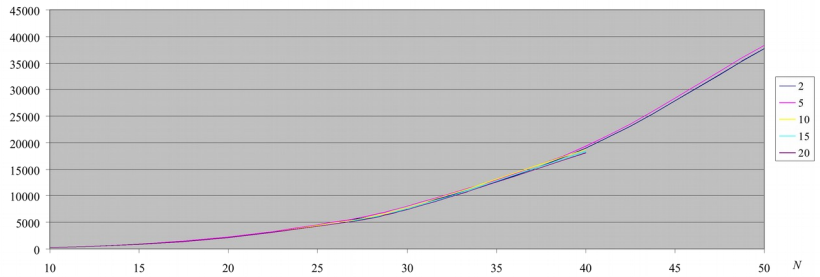


Рис. 3.71. Залежність середньої кількості аналізованих векторів від розмірності задачі при різній кількості обмежень для Max2

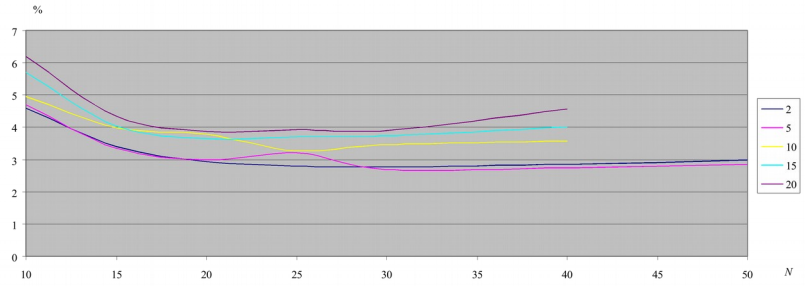


Рис. 3.72. Залежність похибки алгоритму Max5 від розмірності задачі при різній кількості обмежень

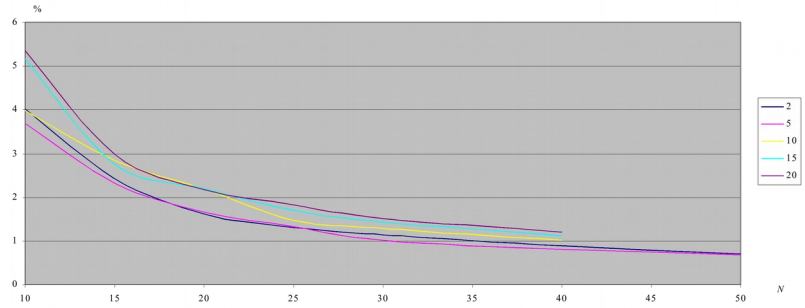


Рис. 3.73. Залежність похибки алгоритму Max2 від розмірності задачі при різній кількості обмежень

У процесі експерименту досліджувалися такі стратегії формування шляхів у графі G_Δ:

стратегія Max2 (при побудові векторів обмежень використовувалася сортування за зменшенням відношення коефіцієнтів функціонала до відповідних коефіцієнтів відповідного обмеження, при цьому при формуванні шляхів наступного рангу на кожному кроці формувалися шляхи, що відповідають найкращому гарантованому прогнозу, і шляхи із кращим оптимістичним негарантованим прогнозом);

стратегія Max4 (при побудові шляху з усіх вершин вибирається краща за прогнозом вершина на ярусі, і вона включається в шлях, що далі випливає з множини вершин, що залишилися, так само);

стратегія Max5 (відмінною рисою від ***max*** є те, що побудова починається не з першого рангу, а з рангу на одиницю менше, ніж значення максимального рангу, на якому будь-який вектор побудованого шляху задовольняє обмеження).

З рисунків видно, що введення гарантованого прогнозу в алгоритми розв'язання одномірної задачі (3.10)–(3.11) на основі процедури А дозволяє знизити похибку з 3 до 0,29 %, що зі збільшенням розмірності асимптотично прагне до 0 і вже при n>100 не перевищує 0,08 %. Останнє обумовлено тим, що, як видно з рис. 3.6.42, зі збільшенням розмірності

задачі кількість неточних розв'язків прагне до 0. Часова складність як точних, так і наближених алгоритмів зменшується на 40 % і більше відсотків (див. рис. 3.68 – 3.69). У випадку розв'язання багатомірної задачі (3.1)–(3.3) найкращою стратегією є стратегія Max2, у якої часова складність практично не залежить від кількості обмежень, а похибка асимптотично прагне до нуля й при n>40 не перевищує 1 %. Середнє значення часової складності аналізованих алгоритмів з довірчою ймовірністю 0,95 не перевищує $O(cn^3)$, де $c \le 1$.

3.7. Задачі динамічного управління в телекомунікаційних мережах

3.7.1. Задача розподілу зон управління в телекомунікаційних мережах

Забезпечення максимальної функціональної потужності мережі при заданій оперативності управління мережею й заданою оперативністю розв'язання ІРЗ у мережі вимагає поєднання децентралізованого управління розв'язанням задач динамічного управління в мережі з централізованим управлінням розв'язанням ІРЗ у мережі. Функції адміністратора в мережі для підвищення відмовостійкого функціонування мереж повинні бути розподілені між декількома пунктами управління мережею, при цьому мережа розбивається на K зон (доменів) [1–25] і в кожній зоні управління розміщаються управляючі сервери, які здійснюють управління кожний у своєму домені й обмінюються службовою інформацією між собою. Критерієм вибору місця положення пунктів управління мережі є мінімум тривалості циклу управління, обумовленого часом збору інформації про стан об'єктів і доведення до них управляючої інформації. Таким чином, виникає задача розподілу пунктів управління в мережі, що забезпечує мінімальний сумарний час збору інформації про стан об'єктів мережі й доведення управляючої інформації до об'єктів мережі. У мережі, описуваній графом $G(V, E)$, виділимо підграф $G`(V`, E`)$, що визначає вузли мережі, які можуть виконувати функції управляючого сервера в мережі.

Визначення плану розподілу зон управління в мережі

Таким чином, із множини вузлів $V`$ необхідно знайти або мінімальну кількість вузлів, яка б охопила управлінням всю мережу, або підмножину вузлів, що мінімізує цикл управління в мережі в умовах впливу перешкод. Тобто задача повинна розв'язуватися з урахуванням деградації мережі в реальному масштабі часу. Для кожної вершини графа $k \in V`$ відповідно до

$G(V,E)$ можна визначити множину абонентів $\{A_i\}$, які можуть бути опитані за час $t \le t^{доп}$, будуючи дерева найкоротших шляхів з вершин $k \in$ V (на основі алгоритмів, наведених у розд. 4) і при цьому всі абоненти і, для яких час опитування $t_i \ge t^{доп}$, виключаються з підмножини $\{A_i\}$. Уведемо змінні β_{ij} і X_j, обумовлені так:

$$\beta_{ij} = \begin{cases} 1, \text{ якщо } i\text{-й абонент може бути опитаний за час } t \le t^{доп} \text{ з } j; \\ 0, \text{ в іншому випадку;} \end{cases}$$

$$x_j = \begin{cases} 1, \text{ якщо } j\text{-й вузол використовується у якості сервера;} \\ 0, \text{ в іншому випадку.} \end{cases}$$

Тоді розглянуту задачу можна сформулювати так.

Мінімізувати цільову функцію

$$L = \sum_{j=1}^{M} t_j x_j , \tag{3.82}$$

задовольняючим обмеження

$$\sum_{j=1}^{M} \beta_{ij} x_j \ge 1, \quad i = \overline{1, A}, \tag{3.83}$$

$$x_j \in \{0,1\}, \quad t_j \ge 0 . \tag{3.84}$$

В окремому випадку, коли $t_j = 1$, ми приходимо до задачі визначення мінімальної кількості серверів, що охоплюють управлінням всю мережу та в зоні j-го сервера виконується нерівність $t_j \le t^{доп}$ $j = (\overline{1, J})$, де $J = |V|$.

Алгоритм розподілу зон управління в мережі

Крок 1. Використовуючи алгоритм (розд. 4) для вершин $j = (\overline{1, J})$, де $J = |V|$, будуємо дерева найкоротших шляхів і, використовуючи нерівність $t_j \le t^{доп}$, визначаємо підмножини абонентів мережі $\{A_i\}$, які можуть бути опитані за час, що не перевищує $t^{доп}$.

Крок 2. На основі $\{A_i\}$ формуємо матрицю $|\beta_{ij}|$ і вектор $\overline{t_j}$.

Крок 3. Розв'язуємо задачу «про мінімальне покриття» з вихідною матрицею $|\beta_{ij}|$ і вектором $\overline{t_j}$ на основі точного або наближеного алгоритму, покроковий опис яких наведено в розд. 2 (вибір алгоритму розв'язання задачі може визначатися оператором мережі залежно від вимог до оперативності розв'язання задач або задаватися автоматично залежно від режиму функціонування мережі).

Крок 4. Видача управляючих впливів у мережу.

3.7.2. Оптимальне планування розподілу задач у мережі, що забезпечує відмовостійке функціонування телекомунікаційних мереж

Для оцінки процесу управління розподілом задач у мережі будемо використовувати такі показники якості:

– коефіцієнт функціональної потужності мережі, який можна записати у вигляді

$$E_V(\vartheta_\mu^i) = \sum_{i=1}^{n_V} \sum_{\mu=1}^{m_i} \beta_{\mu\ i}, \qquad (3.85)$$

де n_v – загальна кількість ПМ у мережі в стані S_v;

m_i – загальна кількість задач, які здатний розв'язувати i-й ПМ у стані S_v;

$\beta_{\mu i}$ – вага μ-ї задачі i-го ПМ, що характеризує її важливість (пріоритет) для управляючої системи;

ϑ_μ^i – сумарний час доступу до даних у мережі, обумовлений розміщенням сегментів БД мережі на її фізичній структурі

$$T = \sum_{i=1}^{m} \sum_{j=1}^{n} K_j p_i \tau_{ij} x_{ij}, \qquad (3.86)$$

де $\tau_{ij} = \frac{V_i^{(b)}}{\lambda_{ij}}$ – середній час вибору 1 кілобайта інформації з i-го фрагмента у вузлі j;

λ_{ij} – коефіцієнт, що враховує швидкість вибору й обробки 1 мегабайта даних при звертанні до i-го фрагмента БД в j вузлі;

$V_i^{(b)}$ – об'єм зовнішньої пам'яті, необхідний для розміщення i-го фрагмента БД;

p_i – характеристика частоти використання і-го фрагмента БД при функціонуванні СУБД;

K_j – коефіцієнт, що показує швидкість доступу до даних на j-му вузлі мережі;

$x_{ij} = \begin{cases} 1, & \textit{якщо фрагмент БД з номером } i \textit{ розташований у } j \textit{ вузлі мережі}; \\ 0, & \textit{в іншому випадку}; \end{cases}$ – середній час відновлення мережі $T_в$ (методика визначення $T_в$ буде розглянута нижче).

Нехай є M_i, $i = \overline{(1,n)}$ -управляючих ПМ, що служать для управління об'єктами O_ε. Відмови системи зв'язку й відмови ПМ уважаємо незалежними. Під станом $s(t)$ у момент часу t розуміють набір станів усіх модулів у цей момент, тобто $s(t) = \sigma_1,\ldots,\sigma_n$, де $\sigma_i \in \{0, 1\}$:

$$\sigma_i = \begin{cases} 1, & \textit{якщо } M_i \textit{ - модуль, що відмовив } (o\text{ - }ПМ); \\ 0, & \textit{якщо } M_i \textit{ - працездатний модуль } (p\text{ - }ПМ). \end{cases}$$

Початковий стан системи $s(t = 0) = s^0 = 00...0.…0$. Нехай Λ_n – множина усіх станів системи; $D = \{M_1, \ldots, M_n\}$ – множина усіх модулів системи; $\Omega^v = \{U_1, \ldots, U_L\}$ – множина усіх задач, розв'язуваних у мережі в стані S_v; $-\Omega_i^v = \{U_1^i, ..., U_\mu^i, ..., U_m^i\}$ – підмножина задач, які працездатний модуль M_i здатний розв'язувати, коли система перебуває в стані S_v. Для стану s^0 задана множина $\Omega^0 = \{U_j\}$ і початковий розподіл задач між всіма модулями, тобто підмножини $\Omega_i^v = \{U_1^i, ..., U_\mu^i, ..., U_m^i\}$. Нехай $D^f{}_v = \{M_i\}^v$, $D^r{}_v = \{M_i\}^v$ – множини відповідно тих, що відмовили, і працездатних ПМ, що відповідають стану S_v; $A^f{}_v$ – множина усіх власних задач модулів, що відмовили, для стану S_v; $A^r{}_v = \Omega^0 \backslash A^f{}_v$ – множина усіх власних задач р-ПМ для стану S_v. Кожна задача $U_\mu^i \in \Omega_i^v$ характеризується ступенем важливості, що визначає мету функціонування системи управління й задається ваговим коефіцієнтом $\{\beta_{\mu i}\}$. Тоді результат перерозподілу задач визначається корисним ефектом E, оцінюваним сумарною ваговою характеристикою множини задач у даному S_v стані (3.82), називаним функціональною потужністю мережі (E_v). Крім того, кожна задача має ряд інших параметрів. Наприклад, середній час обслуговування (перебування) задачі μ в i-му ПМ, що задається матрицею $||\Delta T_{\mu i}||$ $\mu = \overline{(1, m_i}$; $i = \overline{1, n_v)}$; час пересилання (затримки) $||t_{\mu i}||$ повідомлень задачі μ в i-му ПМ і т. д. Тоді формальна модель задачі перерозподілу набуде такого вигляду. Необхідно забезпечити корисний максимальний ефект перерозподілу

$$E_v = \sum_{i=1}^{n_v} \sum_{\mu=1}^{m_i} \beta_{\mu i} X_{\mu i} \to \max \tag{3.87}$$

при обмеженнях

$$\sum_{i=1}^{n_v} \sum_{\mu=1}^{m_i} \Delta T_{\mu i} X_{\mu i} \le \Delta T_v^{доп}; \tag{3.88}$$

$$\sum_{i=1}^{n_v} \sum_{\mu=1}^{m_i} V_{\mu i} X_{\mu i} \le V_v^{доп}; \tag{3.89}$$

$$\sum_{i=1}^{n_v} \sum_{\mu=1}^{m_i} t_{\mu i} X_{\mu i} \le T_v^{доп}; \tag{3.90}$$

$$\sum_{\mu=1}^{m_i} X_{\mu i} \le 1, \tag{3.91}$$

де

$$X_{\mu i} = \begin{cases} 1, & якщо\ \mu\text{-}задача\ розв'язується\ у\ i\text{-}му\ ПМ; \\ 0, & в\ іншому\ випадку; \end{cases} \tag{3.92}$$

$\Delta T_v^{доп}$, $V_v^{доп}$, $T_v^{доп}$ – припустимі граничні значення відповідно сумарного середнього часу обслуговування задачі в i-му ПМ, об'єму пам'яті, необхідного для розв'язання задачі і, і часу перерозподілу інформації у всій мережі в цілому (визначається циклом відновлення

інформації в мережі).

Обмеження (3.92) визначає, що задача може бути призначена для розв’язання тільки на один вузол. Таким чином, задача перерозподілу (3.87)–(3.92) зводиться до багаторазового розв’язання задачі ЦЛП із БЗ в умовах деградації мережі, яку необхідно розв’язувати в масштабі реального часу. Оскільки задача (3.87)–(3.92) при виникненні відмов у мережі повинна розв’язуватися в комплексі з задачами реконфігурації бази дані мережі й відновлення мережі, розглянемо математичні моделі кожної із цих задач окремо й на їхній основі побудуємо загальний алгоритм розв’язання розглянутих задач.

3.7.3. Задача відображення фрагментів бази даних мережі на її фізичну структуру й оптимальний пошук інформації в ній в умовах деградації мережі

Виникнення відмов у мережі, пов'язаних з виходом з ладу елементів зовнішньої пам'яті, що містять сегменти БД мережі, можуть призвести до повної відмови мережі, тому частину найбільш важливих фрагментів дублюють по різних вузлах мережі. Якщо кількість різних фрагментів бази дані мережі позначити через m, то для нормального функціонування мережі повинна виконуватися нерівність

$$\sum_{i=1}^{m}\sum_{j=1}^{n} X_{ij} \geq m,$$

де $x_{ij} = \begin{cases} 1, & \text{якщо фрагмент БД з номером } i \text{ розташований у вузлі } j \text{ мережі}; \\ 0, & \text{в іншому випадку.} \end{cases}$

Якщо нерівність не виконується, то виникає необхідність у перерозподілі фрагментів БД на нову фізичну структуру мережі, що виникла в результаті відмов функціональних елементів, мережі або їх відновлення, при цьому показником якості перерозподілу фрагментів є сумарний час доступу до мережі [246, 247, 270, 271]. З урахуванням уведених у п. 3.7.2 позначень математична модель розподіленої структури БД набуває такого вигляду:

$$T = \sum_{i=1}^{m}\sum_{j=1}^{n} K_j p_i \tau_{ij} x_{ij} \rightarrow \min \qquad (3.93)$$

при обмеженнях

$$\sum_{i=1}^{m} V_i^{(b)} X_{ij} \leq V_j^{(b)}, j = \overline{(1,n)}, \qquad (3.94)$$

$$\sum_{i=1}^{m} V_i^0 X_{ij} \le V_j^{(0)}, \tag{3.95}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} t_{ij} X_{ij} \le T_\partial, \tag{3.96}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} X_{ij} \ge m, \tag{3.97}$$

де V^0_j – об'єм доступної для СУБД на вузлі j оперативної пам'яті;

t_{ij} – середній час передачі даних по еКЗплуатованих каналах зв'язку;

T_∂ – припустимий сумарний час передачі даних по мережі.

У БД важливо організувати ефективний пошук інформації з погляду мінімізації часу її пошуку. Припустимо, що m записів зберігається в n масивах інформації довжини $C_j \; j = \overline{(1,n)}$ і потрібно знайти всі m записів і при цьому провести перегляд масивів якомога меншої довжини. Складемо матрицю $A = \|\alpha_{ij}\|$, у якій стовпці відповідають масивам, а рядки відповідають записам, що містяться в масивах. Елемент α_{ij} дорівнює 1, якщо в j-му масиві міститься i-й запис, і 0 в іншому випадку. Тоді нам потрібно знайти мінімальну кількість масивів, у яких містяться m записів, якщо масиви однакової довжини, або кількість масивів мінімальної довжини, у яких містяться всі m записів у випадку масивів різної довжини, що дозволить переглядати не всі масиви БД, кількість яких у БД може бути досить великою, і за рахунок цього зменшити час пошуку інформації. Ця задача відома як задача про мінімальне зважене покриття й у цьому випадку вона має вигляд

$$\min \sum_{j=1}^{n} C_j X_j \tag{3.98}$$

при обмеженнях

$$\sum_{j=1}^{n} \alpha_{ij} x_{ij} \ge 1, \quad i = \overline{(1,m)}; \quad j = \overline{(1,n)}; \quad x_j \in \{0,1\}, \tag{3.99}$$

$$\alpha_{ij} = \begin{cases} 1, & \textit{якщо рядок } i \textit{ покривається стовпцем } x_j; \\ 0, & \textit{в іншому випадку,} \end{cases}$$

де n – кількість масивів;

m – кількість записів, які необхідно знайти в n масивах;

c_j – довжина j-го масиву.

Розв’язання задачі (3.98)–(3.99) на основі рангового підходу розглянуто в п. 3.5.

3.7.4. Оптимальне планування відновлення телекомунікаційної мережі при відмовах її функціональних елементів

Нехай у процесі функціонування мережі в моменти часу t_1, t_2 ,…, t_i у ПМ виникають пакети відмов $\{y_j\}$ $j=\overline{(1,K)}$, і для усунення кожного пакета відмов $\{y_j\}$ потрібне виконання комплексу робіт P^j, реалізація якого можлива за наявності множини $\{X_i^j\}$-необхідних засобів і фахівців. Множину $\{X_i^j\}$ можна подати у вигляді об'єднання двох множин

$$X_H^j = \alpha_h^j \cup \vartheta_l^j, \quad h=\{\overline{1,H}\}; \quad l=\{\overline{1,L}\}, \tag{3.100}$$

де α_h^j – множина засобів h–го типу, необхідних для виконання комплексу робіт P^j;

ϑ_l^j – множина фахівців l–ї спеціальності, які можуть забезпечити виконання комплексу робіт P^j.

Позначимо час відновлення, пов'язаний з усуненням $\{y_j\}$ пакета несправностей – $t_в^j$. Він визначається:

1) часом $t_{кр}^j$ виконання робіт, що лежать на критичному шляху $\mu_{кр}^j$ у сітковому графіку усунення j-ї відмови

$$t_{кр}^j = \sum_{q\in\mu_{кр}} t_q^j, \tag{3.101}$$

де t_q^j – час виконання q-ї критичної роботи P_q^j з усунення j-ї відмови;

2) часом очікування $t_{ож}^j$, що залежить:

а) від часу виявлення $t_{обн}^{jj}$-ї відмови;

б) часу ухвалення рішення $t_{пр}^j$ на формування оптимального плану усунення відмов;

в) часу доставки $t_д^j$ виділених відповідно до плану $\ddot{I}_r^j(t_i)$ засобів $X^j(t_i)$ у місця призначення для здійснення комплексу робіт P^j.

Таким чином, час очікування визначається як

$$t_{ож}^j = \begin{cases} t_{обн}^j + t_{пр}^j + t_д^j, \text{якщо} \quad X_j(t_i) \ge X_н^j(t_i); \\ t_{обн}^j + t_{пр}^j + t_д^j + \Delta t_r^j(t_i) - \text{в} \quad \text{іншому} \quad \text{випадку}, \end{cases} \tag{3.102}$$

де Δt_r^j – затримка у виконанні комплексу робіт P^j через невиконання умови $X^j(t_i) \ge X_н^j(t_i)$.

Єдиний показник, на який ми можемо впливати за рахунок оптимального планування, може полягати в розробленні послідовності планів розподілів

$$П_r(t_1), П_r(t_2), \ldots, П_r(t_i), \tag{3.103}$$

що забезпечують максимальну функціональну важливість виконання всього комплексу робіт. Породження планів $П_r(t_i)$ здійснюється в процесі багаторазового розв'язання такої задачі:

$$\sum_{j=1}^{K} b_j X_j \to \max \tag{3.104}$$

при обмеженнях:

а) на кількість необхідних засобів h–го типу для відновлення j-го пакета відмов

$$\sum_{j=1}^{K} \alpha_{hj} X_j \le \alpha_{h0}, \quad h = \overline{(1, H)}; \tag{3.105}$$

б) на кількість фахівців l-ї спеціальності для відновлення j-го пакета відмов

$$\sum_{j=1}^{K} \vartheta_{lj} X_j \le \vartheta_{l0}, \quad l = \overline{(1, L)}, \tag{3.106}$$

де $X_j = \begin{cases} 1, & якщо \ (\alpha_{hj} \ и \ \vartheta_{lj}) \ включений \ у \ план \ П_r(t_l); \\ 0, & в \ іншому \ випадку. \end{cases}$

b_j – сумарна важливість задач U_μ, які не можуть бути розв'язанні в *о*–ПМ у результаті *j*-ї відмови, тобто

$$b_j = \sum_{U_\mu \in A_j^r} \beta_\mu . \tag{3.107}$$

За отриманим планом $П_r(t_i)$ знаходимо час очікування $t_{ож}$ групи в цілому, тобто

$$t_{ож}(П_r(t_i) = \sum_{j=1}^{K} t_{ож}^j X_j . \tag{3.108}$$

Кожна робота P^j, що входить в оптимальний план $П_r(t_i)$, згідно з виразом (3.101) характеризується часом $t_{кр}^j$ виконання робіт, що лежать на критичному шляху $\mu_{кр}^j$ у сітковому графіку усунення j–ї відмови. Тому сумарний час $t_{кр}^j$ критичних шляхів у групі дорівнює

$$t_{кр}(П_r(t_i) = \sum_{j=1}^{K} t_{кр}^j X_j . \tag{3.109}$$

Тоді час відновлення $T_в^*$ усього комплексу робіт з урахуванням наявних ресурсів буде визначатися співвідношенням

$$T_в^* = t_{кр} + t_{ож}. \qquad (3.110)$$

У випадку, коли у БР передбачені резервні засоби, час переходу на резерв може бути врахований у часі відновлення. Крім того, якщо вузли, що вийшли з ладу, не відновлюються, то необхідно покласти $Tв = \infty$.

3.7.5. Оптимальне планування передислокації пунктів управління в нестаціонарній телекомунікаційній мережі

Задача розміщення ПУ мережі виникає в мережах з мобільними абонентами, при цьому координати абонентів мережі визначаються вимогами, пропонованими до якості розв’язання ІРЗ, розв'язуваних у мережі з мобільними абонентами. Нехай мобільний абонент перебуває в точці з координатами φ, β. Імовірність того, що він встановить зв'язок з ПУ y_i, дорівнює $P(d_i)$, де d_i – відстань від точки з координатами φ, β до точки розташування y_i з координатами α_i, η_i. Величина $P(d_i)$ може розглядатися і як імовірність зв'язку з урахуванням рельєфу конкретної траси між зазначеними точками. Завдання полягає у визначенні мінімальної кількості ПУ й місць їхнього розміщення для забезпечення стійкого зв'язку. Для цього територію позиційного району, схема якого подана на рис. 3.74, представимо у вигляді матриці $A = \|a_{ij}\|$ з М рядками й N стовпцями, у якій елементи a_{ij} набувають значення, що дорівнює 1, якщо ПУ, установлений на j позиції, може забезпечити виконання умови $P_{ij} \geq P_д$ (припустиме значення ймовірності зв'язку), при цьому будемо говорити, що ПУ, встановлений на j позиції, покриває i квадрат, і $a_{ij} = 0$ – в іншому випадку.

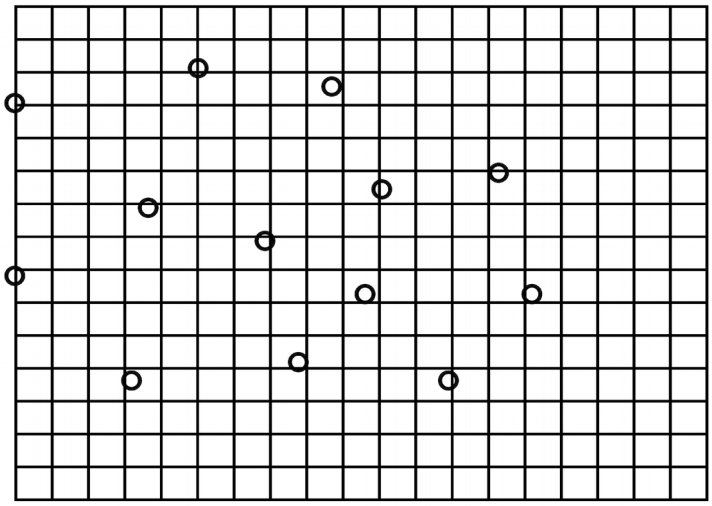


Рис. 3.74. Схема розміщення ПУ в позиційному районі

У загальному випадку задача оптимального розміщення ПУ зводиться до пошуку мінімального покриття усіх рядків у матриці А множиною стовпців, тобто необхідно знайти вектор x^*, що мінімізує цільову функцію

$$L = \sum_{j=1}^{M} c_j x_j \to \min,$$

і задовольняє обмеження

$$\sum_{j=1}^{M} a_{ij} x_j \geq 1, \quad i = \overline{1, A}, \; x_j \in \{0,1\}, \quad c_j \geq 0,$$

де $a_{ij} = \begin{cases} 1, \text{коли } i \text{ змінна може бути покрита;} \\ 0 \text{ - в іншому випадку.} \end{cases}$

Коефіцієнти цільової функції c_j характеризують ступінь пристосованості позиції для розміщення на ній ПУ. Зокрема, як такий ваговий коефіцієнт також може вибиратися час зайняття позиції пунктом управління. Результатом розв'язання цих задач буде множина позицій, на яких необхідно розмістити ПУ. Для того ж щоб визначити, який ПУ й на якій позиції повинен бути встановлений, а також для оцінки доцільності маневру, необхідно розв'язати транспортну задачу з мінімізацією максимального часу висування ПУ на позиції з місць їхньої дислокації. Для розв'язання цих задач можуть бути застосовані алгоритми переборного типу, однак перебір усіх варіантів розміщення зажадає аналізу $C_M^m \times m! = \frac{M!}{(M - m)!}$ комбінацій. Для зменшення їхньої кількості й зниження,

тим самим, часової складності розв'язуваних задач можна звести дану задачу до задачі про найменше покриття, методи розв’язання якої наведені в розд. 2 даного посібника. Слід зазначити, що у випадку, коли c_j=1, для усіх j у розглянутій задачі визначається мінімальна кількість ПУ, що забезпечується заданою кількістю мобільних абонентів мережі.

3.7.6. Управління виконанням завдань у вузлах телекомунікаційної мережі

Забезпечення необхідної оперативності розв’язання задач управління в мережі й заданої оперативності розв’язання ІРЗ вимагає використання багатопроцесорних систем у вузлах розв’язання мереж, для забезпечення відмовостійкого функціонування яких потрібне розв’язання задачі розподілу ресурсів у реальному масштабі часу, обумовленому циклом відновлення інформації на мережі. Таким чином, будемо розглядати і–й вузол мережі як БПС, що складається з n ПМ, що мають свою локальну пам'ять. Дані, необхідні для розв’язання задач, і розв'язувані задачі вводяться в локальну пам'ять процесорних елементів. Всі задачі початкової множини $\Phi\{u_j\}\quad j=(\overline{1,L})$ зберігаються в загальній пам'яті БПС і при відмовах ПМ здійснюється завантаження задач у пам’ять працездатних ПМ відповідно до знайденого плану розподілу (рис. 3.75).

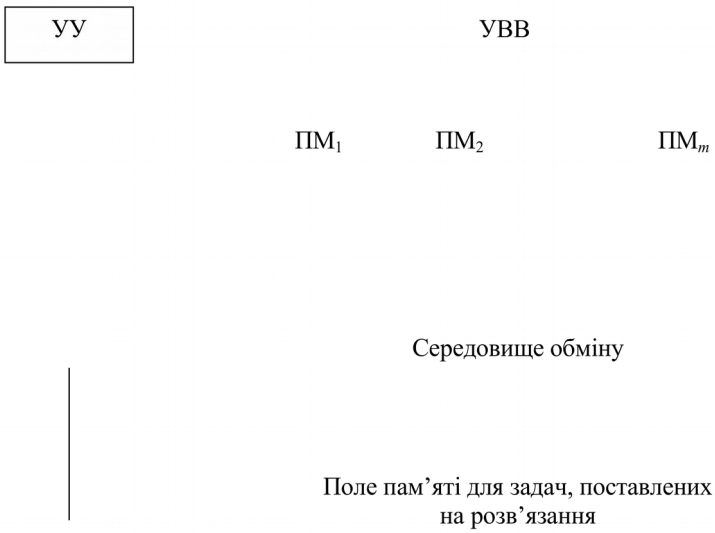


Рис. 3.75. Структура обчислювача у вузлі мережі

Стан системи визначимо як двійковий набір $\rho=(\sigma_1, \sigma_2, ..., \sigma_n)$, де σ_i = 0 для працездатних ПМ і σ_i = 1 для непрацездатних ПМ. Кількість можливих станів дорівнює 2^n. Ефективність E_v БПС у стані s_v будемо характеризувати множиною задач, які може розв’язувати обчислювальна система в цьому стані. Кожна задача характеризується ступенем

важливості, що визначає мету функціонування системи й задається ваговими коефіцієнтами

$$E_v = \sum_{U_j \in \Phi_v} \beta_j .$$

У загальному випадку $\{\beta_j\}$ можуть коректуватися операторами мережі з урахуванням циклу відновлення інформації в мережі для більш адекватної відповідності цілей, які переслідуються при розв'язанні підмножини задач на даній обчислювальній системі, загальним цілям, які повинна реалізовувати система управління, де дана обчислювальна система використовується.

Будемо вважати, що відомо час τ_j розв'язання задачі j, об'єм пам'яті V_j, необхідний для зберігання програм і даних, E^*– мінімально припустиме значення E_v, Q – мінімальний рівень відмовостійкості, T_∂ – мінімально припустимий час однократного розв'язання задачі. Уважаємо, що $\tau_j \le T_\partial$, $V_j \le V_\partial$, де V_∂ – припустимий об'єм пам'яті, займаний задачею на усіх ПМ, тобто будь–яка задача $u_j \in \Phi$ може бути розв'язана в будь-якому ПМ. Для визначення необхідного рівня відмовостійкості БПС Q^* необхідно:

1) визначити граничну кількість q–процесорних модулів $m \ge q$, при якій система залишається працездатною;

2) для кожного стану, що містить кількість працездатних ПМ $m \ge q$, знайти такий план розподілу задач, який би забезпечував максимальне значення сумарного коефіцієнта важливості за наявності обмежень на час розв'язання задач і об'єм пам'яті, займаний задачами й вихідними даними для їхнього розв'язання.

Якщо кількість процесорних елементів досить велика й розв'язання розподілу задач може виявитися дуже трудомістким, тоді доцільно знайти план розподілу, що забезпечує максимальне значення сумарного коефіцієнта важливості за наявності обмежень на час розв'язання задач і об'єм пам'яті, займаний задачами й вихідними даними для їхнього розв'язання. Якщо кількість процесорних елементів досить велика й розв'язання розподілу задач може виявитися дуже трудомістким, тоді доцільно знайти план розподілу, що забезпечує максимальне значення сумарного коефіцієнта важливості розв'язуваних задач не для усіх станів s_v, а для обмеженої кількості q працездатних ПМ. У цьому випадку для довільного стану s_v розглянута задача може бути зведена до такої задачі булевого програмування:

$$E^q = \sum_{j=1}^{L} \sum_{i=1}^{q} \beta_j X_{ij} \to \max \qquad (3.111)$$

при обмеженнях

$$\sum_{j=1}^{L} t_j X_{ij} \le T_\partial \quad , i = (\overline{1,q}), \tag{3.112}$$

$$\sum_{j=1}^{L} V_j X_{ij} \le V_\partial \quad , i = (\overline{1,q}), \tag{3.113}$$

$$\sum_{j=1}^{L} X_{ij} \le 1, \tag{3.114}$$

$X_{ij} = \begin{cases} 1, & \text{якщо задача } j \text{ розв'язується на } i\text{-му ПМ}; \\ 0, & \text{в іншому випадку.} \end{cases}$

Обмеження (3.114) означає, що задача j розв’язується на одному ПМ.

Таким чином, алгоритм перерозподілу задач між ПМ вузла мережі складається з таких кроків:

Крок 1. Перевіряємо, чи з'явилися модулі, що відмовили, у вузлі мережі, якщо ні, то вузол не змінює режиму роботи, якщо так, то виконуємо наступний крок.

Крок 2. Розв'язуємо задачу (3.111)–(3.113) алгоритмом A_m або B_m розд. 2 і перевіряємо поточне значення $E^q < E^*$, якщо нерівність виконується, то на управляючий сервер свого домену вузол видає сигнал про функціональну відмову вузла, в іншому випадку переходимо до виконання наступного кроку.

Крок 3. На основі оптимального розв'язку задачі (3.111)–(3.113) визначаються задачі, що залишаються в черзі на розв’язання у вузлі, й інформація про ПМ, що відмовили, передається на управляючий сервер свого домену, після цього здійснюється перехід до кроку 1.

3.7.7. Управління потоками інформації й маршрутизацією в телекомунікаційних мережах

Як зазначалося раніше, особливістю управління потоками інформації в мережах спеціального призначення є необхідність забезпечення заданої прихованості передачі інформації між абонентами мережі, при цьому вимоги до прихованості передачі інформації можуть висуватися або до всієї мережі в цілому, або тільки до деякої підмножини абонентів, обумовленої адміністратором мережі на даний момент часу. Кожній передачі повідомлень від вузла i до вузла j приписується деяка ймовірність P_{ij} перехоплення повідомлень. Із систем діагностики в циклі відновлення інформації в мережах уводяться дані про зміну матриць $|P_{ij}|$. Очевидно, що шляхи передачі повідомлень у мережі повинні утворювати остове дерево графа G мережі або підграфа $G^\wedge \subseteq G$ заданою адміністратором мережі для обміну інформацією між абонентами мережі. І потрібно знайти таке остове дерево, що мінімізує величину $1 - \Pi\,(1 - P_{ij})$. Тут добуток береться по тих ребрах, що утворять це дерево. Як показано в роботі [161],

ця функція зростаюча й симетрична відносно P_{ij}, і шукане дерево буде збігатися з найкоротшим остовним деревом графа G або $G^{\wedge}$, заданим адміністратором мережі, при цьому P_{ij} приймаються за "вартості" ребер графа C_{ij}. Розглянемо паралельний алгоритм НОД графа на основі рангового підходу.

В основу алгоритму покладена така послідовна процедура побудови найкоротшого шляху, що являє собою досить економічний "поглинаючий" алгоритм [178]. Алгоритм починає роботу з вибору довільної вершини графа й найкоротшого ребра із множини ребер, що з'єднують цю вершину з іншими вершинами. З'єднаємо дві вершини обраним ребром. Виберемо найближчу до цих вершин третю вершину. Додамо цю вершину й відповідне ребро до отриманого графа. Продовжуємо даний процес доти, поки всі вершини не будуть з'єднані між собою. Процедура, заснована на "поглинанні" найкоротших дуг, може бути описана в такий спосіб.

Процедура побудови НОД

Крок 1. Використовуючи вершини вихідного графа G, визначаємо такі дві множини: S – множина з'єднаних вершин; $\overline{S}$ – множина нез'єднаних вершин. На початку всі вершини будуть належати множині $\overline{S}$.

Крок 2. Вибрати довільну вершину з $\overline{S}$ і з'єднати її з найближчою сусідньою вершиною.

Крок 3. Серед усіх ребер, що з'єднують вершини з множини S з вершинами з множини $\overline{S}$, вибрати найкоротше ребро. Кінцеву вершину цього ребра, що лежить в $\overline{S}$, позначимо δ. Видалити вершину δ із множини $\overline{S}$ й помістити її в множину S.

Крок 4. Виконувати крок 3 доти, поки всі вершини не будуть належати множині S.

На основі розробленої в роботі [79] процедури пошуку K запропонуємо такий паралельний алгоритм визначення K, заснований на переході від аналізу графа G до його стягнутого дерева шляхів від довільної вершини s графа G. Ребро $s - i \in M_i^{r=k}$ належіть множині ребер, інцидентних вершині i на k-му кроці роботи процедури побудови *НОД*. На кожному кроці роботи будуть одночасно формуватися ребра i – j у кожній з підмножин M^r, при цьому побудова *КО* здійснюється в такий спосіб.

Паралельний алгоритм визначення НОД

Крок 1. Відповідно до матриці суміжності в підмножинах M^r_i формуються ребра $s - 1$ ($i = \overline{1, n}$).

Крок 2. Серед множини ребер $s - i \in M_i^r$ вибираємо ребро (q^*, p) з мінімальною вагою й заносимо в масив W, надалі множину M_p^r виключаємо з розгляду.

Крок 3. На базі вершини *P* у підмножинах M_i^r, для яких $i \neq q$, формуємо ребра *P–i* (*i*=$1,\overline{n}$) і в них же відновлюємо ребра (1–i), сформовані на попередньому кроці в M_i^r ребра мінімальної ваги. Якщо на попередньому кроці виявилося кілька ребер мінімальної ваги, що кінчаються на різних вершинах, то на базі цих вершин формуються всі можливі ребра в множинах M_i^r і відновлюються ребра з мінімальною вагою, отримані на попередньому кроці.

Крок 4. Перевіряємо, чи всі підмножини M_i^r виключені з розгляду, якщо так, то алгоритм закінчує роботу, оскільки множина ребер масиву *W* утворять НОД; якщо ні, то переходимо до виконання кроку 2.

Становить інтерес оцінка складності реалізації даного алгоритму на *n*–процесорах і його порівняння з одним з найефективніших паралельних алгоритмів SOLLIN з тимчасовою складністю *O*(flog *n*) [79].

Оцінку складності розробленого паралельного варіанта спочатку зробимо для його послідовної реалізації. У множині Мі максимальна кількість сформованих ребер не може перевищити (*n* – 1) і, отже, виділення найкоротшого ребра в *Mi* зажадає *O*(log(*n* – 1)) часу. Після виділення найкоротших ребер у {*Mi*} для визначення найкоротшого ребра буде потрібно *O*(log *n*) часу. Отже загальна часова складність одного кроку, на якому формуються Мі і визначаються вершини, на базі яких будуть формуватися ребра в підмножинах Мі наступного рангу, не може перевищити *O*(*n* + (*n* – 1)log(*n* – 1) + log *n*) = *O*(*n* + nlog *n*). Оскільки на n–процесорній системі кожний процесор переглядає одночасно елементи свого рядка матриці суміжності, то складність одного такого кроку на *n*–процесорній системі не перевищить *O*(log *n*). Оскільки цих кроків усього повинно бути (*n* – 1), то загальна складність розробленого паралельного алгоритму не перевищить *O* (*n*log *n*).

Таким чином, алгоритмічна складність розв'язання задачі про знаходження НОД, здійснювана на основі рангового підходу, така сама, як і для кращих відомих паралельних алгоритмів. Якщо порівняти розглянутий паралельний алгоритм визначення НОД з паралельними алгоритмами визначення найкоротших шляхів паралельна реалізація алгоритму Дейкстри і знаходження шляхів з максимальною пропускною здатністю, то легко побачити, що вони всі можуть бути реалізовані на одній і тій самій паралельній обчислювальній структурі. Отже, при використанні такого ПОВ адміністратор мережі може задавати показник якості (3.40)–(3.43), на основі якого варто вибирати метод маршрутизації в мережі. Крім визначення оптимальних маршрутів і забезпечення прихованості передачі інформації в мережі необхідно вміти оцінювати й пропускну здатність мережі. Як показано раніше, ця задача зводиться до розв'язання ЗНП, для якої вже розроблено ефективні алгоритми її розв'язання.

3.7.8. Алгоритм функціонування телекомунікаційної мережі

На основі проведеної формалізації комплексу задач динамічного управління, розв'язуваних у мережах, і розглянутих методів їхнього розв’язання пропонується такий алгоритм функціонування мереж.

У початковому стані $s(t = 0) = 00...0$, тобто всі ПМ працездатні. Кількість *р–р*–ПМ *Mi* дорівнює $n_v\, i=(1, n_v)$. Передбачаються відомими час розв’язання $\{t_{рз}\}$ задач, що залежить від величини вихідних параметрів розмірності задачі ($n, m, r_{ср}$). Цей час може бути представлений у вигляді матриці, значення елементів {tрз} якої залежить від конкретного виконання алгоритмів, описаних у розд. 2.

Крок 1. Установлюємо вихідний стан технічних засобів ВР, що полягає: а) у виборі множини задач U для кожного ПМ; б) переведенні усіх обчислювальних засобів ПМ у стан готовності; в) підключенні об'єктів управління до відповідної ПМ. У вихідний стан ВР переводиться у випадках: а) першого запуску при проведенні пусконалагоджувальних робіт; б) після її відновлення у випадку повної відмови; в) при плановому проведенні регламентних робіт. Результатом цього кроку є готовність ВР до розв’язання заданої множини ІРЗ на працездатних ПМ (*р*–ПМ).

Крок 2. Через інтервали часу, задані циклом управління, здійснюється контроль стану. Аналіз управляючою програмою отриманого s_v –стану ВР і порівняння його з попередньою s_{v-1}. Установлення необхідних режимів функціонування мережі. При зміні стану s_v переходимо до виконання наступного кроку, інакше операційна система запам'ятовує поточний стан s_v до наступного циклу управління. Здійснюється перевірка, чи надійшли запити від адміністраторів доменів на перерозподіл зон управління, якщо так, то виконуємо наступний крок, інакше переходимо до виконання кроку 4.

Крок 3. Визначення плану перерозподілу зон управління, для цього розв’язується задача (3.82)–(3.84) на основі алгоритму, наведеного в п. 3.5.1.

Крок 4. Адміністратор мережі перевіряє, чи надійшли запити на передислокацію елементів мережі, якщо так, то виконуємо наступний крок, інакше переходимо до виконання кроку 6.

Крок 5. Знаходимо план передислокації елементів мережі на основі алгоритму, розробленого в п. 3.7.5.

Крок 6. Адміністратор мережі перевіряє, чи відбулися відмови в мережі за даний цикл управління, якщо так, то виконуємо наступний крок, інакше виконуємо крок 1.

Крок 7. Адміністратор мережі перевіряє, чи потрібна реконфігурація БД ($T_{\Sigma} \le T_{ДОП}$), якщо так, то виконуємо наступний крок, інакше переходимо до виконання кроку 15.

Крок 8. Визначається план реконфігурації БД, для цього розв’язується задача (3.84) на основі наближеного або точного алгоритмів

розв'язання задачі ЦЛП із БЗ, розроблених у розд. 2, і з урахуванням отриманого розв'язку перевіряється нерівність $T^*_{\Sigma} \le T_{ДОП}$, якщо нерівність виконується, то переходимо до виконання кроку 9, інакше виконуємо крок 12.

Крок 9. Адміністратор мережі перевіряє, чи змінилися P_{ij}, якщо так, то визначається підмережа, що забезпечує мінімальне значення ймовірності перехоплення інформації супротивником (визначення підмережі здійснюється на основі алгоритму, розглянутого раніше), інакше виконуємо наступний крок.

Крок 10. Здійснюється оцінка пропускної здатності мережі. Для цього виходячи з топології мережі на даний момент часу будується матриця шляхів Р і за нею визначається величина мінімального покриття сукупності шляхів на основі точних або наближених алгоритмів розв'язання ЗНП, розроблених раніше. Якщо поточне значення потоку перевищує величину максимально можливих, обумовлених мінімальним покриттям сукупності шляхів, то приймається рішення про обмеження навантаження в мережі й видаються відповідні управляючі впливи в мережу (для обмеження навантаження можуть використовуватися стандартні процедури й протоколи на транспортному рівні TCP, на міжмережевому рівні стандарт MIL–STD–1781).

Крок 11. Адміністратором мережі задається критерій визначення оптимальних маршрутів (3.39)–(3.41). Виходячи з обраного критерію або на основі алгоритму А п. 3.7.1, або алгоритмів визначення НШ будується дерево оптимальних маршрутів у мережі й видаються відповідні управляючі впливи в мережу для корекції матриць маршрутів і матриць, що характеризують поточний стан мережі, а також можуть видаватися рекомендації з вибору режимів функціонування мережі операторам різних рівнів управління.

Крок 12. Визначається оптимальний план пошуку інформації в мережі й $T^*_{П.}$ Для визначення оптимального плану пошуку й $T^*_{П}$ розв'язується задача (3.98)–(3.99) на основі точного або наближеного алгоритмів розв'язання ЗНП, наведених у розд. 2. Далі перевіряється нерівність $T^*_{П} \le T_{\Sigma}^{ДОП}$, якщо вона виконується, то переходимо в режим оптимального пошуку (розв'язання задачі оптимального пошуку в кожному циклі управління) і виконуємо крок 10, інакше виконуємо наступний крок.

Крок 13. Адміністратор мережі визначає план відновлення мережі, для цього здійснюється розв'язання задачі (3.104)–(3.105) на основі наближених і точних алгоритмів розв'язання задачі 0,1–рюкзак, наведених у розд. 2. Оцінюється час відновлення мережі $T_В$ і перевіряється виконання нерівностей $T^v_{\Sigma} + T_В \le T_{ДОП}$; $T_П + T_В \le T_{ДОП}$, якщо одна з нерівностей не виконується, то це означає, що відбулася відмова мережі, про що доповідається операторам усіх рівнів і здійснюється перехід до виконання

наступного кроку, інакше видається управляючий вплив щодо відновлення мережі й здійснюється перехід до кроку 10.

Крок 14. Адміністратор мережі дає запит вищому рівню про можливість зниження E_0, якщо зниження можливо, то $E_0 = E_0$ і переходимо до виконання наступного кроку, інакше фіксується повна відмова мережі в змісті виконання функцій управління й алгоритм закінчує роботу.

Крок 15. Здійснюється перевірка $E^v \geq E_0$, якщо нерівність виконується, то перевіряємо, чи зменшилося значення E^v, якщо ні, то переходимо до кроку 10, інакше виконуємо наступний крок. Якщо нерівність не виконується, то виконується наступний крок.

Крок 16. Збір поточних параметрів каналів передачі даних, параметрів *p*–ПМ, таких як середній час розв'язання задачі в *R*–му ПМ; витрати ресурсів в *R*–му ПМ для розв'язання задачі. З пам'яті зчитуються коефіцієнти важливості задач, розв'язуваних у групі, які обчислюються заздалегідь, наприклад методом експертних оцінок, і можуть бути скоректовані оператором у процесі функціонування системи. Задаються граничні поточні можливості (як ресурсні, так і часові) самих р–пм на розв'язання ІРЗ. Задається режим обчислень оптимального плану:

режим 1 – швидкий пошук припустимого розв'язання при достатності одержання наближеного розв'язку;

режим 2 – більш повільний пошук точного розв'язання задачі перерозподілу.

Вибір режиму здійснюється оператором з таких міркувань. Наказом на проведення робіт (розрахунків, аналізу результатів і т. д.) задається граничний строк виконання поставленої задачі, у результаті чого на пошук оптимального плану приділяється час $t_{доп}$, а його реалізація займе $t_{рік}$ часу. Якщо $t_{рз}$ (реж. 2)< $t_{доп}$, то ми одержимо точний розв'язок при максимальній ефективності. В інших випадках задача розв'язується наближеними алгоритмами. Далі розв'язується задача виділення такої множини ІРЗ, при якій показник функціональної потужності ВР був би максимальним. У результаті отримуємо оптимальний план перерозподілу, що дає нове значення E^*. Операційна система перевіряє отримане значення $E^* \geq E_0$, якщо нерівність виконана, тобто потрібний перерозподіл задач у мережі, то видається управляючий вплив на пересилання задач у мережі й здійснюється перехід до кроку 9, інакше фіксується відмова вузлів відповідного рівня управління, про що доповідається операторові, а далі перехід до кроку 13.

Оцінимо часову складність алгоритму функціонування мережі. Часові характеристики кроків алгоритму функціонування мережі можна розбити на три приблизно однакових по порядках величин множини. Першу множину складають доданки, що відображають виконання одиночних елементарних операцій (таких як додавання, порівняння), що здійснюється на ЕОМ у мікросекундному діапазоні. До другої множини

доданків належать час збору й задавання різних параметрів: параметри каналів зв'язку або ж припустимий час розв’язання задачі. Це займе час від десятків мілісекунд до декількох секунд. Для першої множини час визначається швидкодією управляючої програми й вузла ВР. Друга множина задається циркулюючими даними в мережі, які оновлюються через однакові інтервали часу, обумовлені циклом відновлення інформації в мережі. Швидкодія третьої множини визначається обраним алгоритмом функціонування мережі. Через складність розв’язання комплексу задач, реалізованих даним алгоритмом, при їхній великій розмірності пошук оптимального розв’язку існуючими методами може призвести до багатогодинних часових витрат. Отже з усіх множин найбільшу трудомісткість має третя, а часовими витратами на інші множини, з погляду розв’язання даного комплексу задач, можна знехтувати. Таким чином, з'являється необхідність розроблення швидкодіючих алгоритмів для того, щоб час пошуку розв’язку був порівняний із часом розв’язання в перших двох групах.

При виконанні даного алгоритму за один цикл управління в найгіршому разі, як це видно з його кроків (1–16), доведеться 4 рази розв’язувати ЗНП 4, 3 рази розв’язувати оптимізаційну задачу ЦЛП із БЗ, 1 раз задачу визначення оптимальних маршрутів за заданим критерієм, 1 раз задачу визначення підмережі, що забезпечує задану прихованість передачі інформації. Як показано в розд. 2, часова складність розв’язання перших двох задач не перевищує $O(n^3m)$, а других двох $O(n^2)$, отже часова складність третьої групи часових показників роботи алгоритму не перевищить $(7*O(n^3m) + 2*O(n^2)) \cong 7*0(n^3m)$. При реалізації даного алгоритму на ПОВ циклічного типу його часова складність може бути знижена в n раз.

3.7.9. Оцінка ефективності розв’язання задач динамічного управління потоками інформації й взаємодією процесів у телекомунікаційних мережах

Спочатку розглянемо імітаційну модель найбільш важливої задачі перерозподілу ІРЗ у мережі.

Оцінка ефективності розв’язання задачі перерозподілу ІРЗ у мережі

Роботу алгоритму функціонування мережі АСУ на кроках розв’язання задачі перерозподілу ІРЗ у ВР проілюструємо на прикладі повнозв’язної мережі з чотирма вузлами. Приймемо такі допущення й обмеження:

1. Вузли ВР називають ЕОК.

2. ЕОК мають у своєму розпорядженні обчислювальні засоби у відповідності зі своєю штатною структурою.

3. На даний момент кожний ЕОК розв’язує однакову кількість однотипних ІРЗ, найменування яких і коефіцієнти важливості, з погляду

цілей функціонування, задаються табл. 3.2, у вигляді матриці $|\beta_{\mu i}|$, де $\mu = (\overline{1,8}); i = (\overline{1,4})$:

$$\beta_{\mu i} = \begin{vmatrix} 4 & 5 & 9 & 37 & 7 & 12 & 23 & 6 \\ 24 & 14 & 18 & 11 & 9 & 18 & 17 & 10 \\ 14 & 12 & 10 & 11 & 15 & 9 & 8 & 10 \\ 9 & 5 & 15 & 61 & 9 & 6 & 5 & 17 \end{vmatrix}.$$

Таблиця 3.2

Вагові характеристики задач

μ	Найменування ІРЗ	$\beta_{\mu 1}$	$\beta_{\mu 2}$	$\beta_{\mu 3}$	$\beta_{\mu 4}$
1	Розрахунок характеристик системи	4	24	14	9
2	Збір і обробка даних про систему	5	14	12	5
3	Збір і обробка даних про систему	9	18	10	15
4	Обробка даних про стан і результати зовнішніх впливів на систему	37	11	11	6
5	Розрахунок і обробка результатів функціонування підсистем	7	9	15	9
6	Розрахунок з планування роботою функціональних підсистем	12	18	19	6
7	Розрахунок перспективних планів розвитку функціональних підсистем	23	17	8	5
8	Обробка даних про матеріальний, технічний, інженерний стани функціональних підсистем	6	10	10	17

Сумарна потужність ВР у стані s_0 згідно з формулою (3.6) дорівнює $E_0 = 395$. При зміні цілей функціонування значення коефіцієнтів $\beta_{\mu i}$ можуть змінюватися. Нехай кожна ІРЗ i–го ЕОК у випадку розв'язання на R-му р-ПМ займає $C^R_{\mu i}$ одиниць ресурсу й вимагає $\Delta T^R_{\mu i}$ часу обслуговування. Тоді для ІРЗ першого ЕОК утворяться матриці $|C^R_{\mu 1}|$ і $|\Delta T^R_{\mu 1}|$:

$$C^R_{\mu i} = \begin{vmatrix} 35 & 40 & 14 & 33 & 21 & 17 & 28 & 46 \\ 31 & 12 & 46 & 34 & 16 & 22 & 16 & 33 \\ 40 & 20 & 33 & 15 & 17 & 17 & 14 & 21 \\ 48 & 25 & 16 & 47 & 38 & 34 & 41 & 17 \end{vmatrix}, \quad \Delta T^R_{\mu i} = \begin{vmatrix} 43 & 28 & 13 & 16 & 13 & 26 & 26 & 39 \\ 36 & 47 & 24 & 17 & 45 & 14 & 44 & 20 \\ 47 & 47 & 20 & 40 & 48 & 41 & 32 & 38 \\ 37 & 34 & 21 & 13 & 29 & 26 & 10 & 21 \end{vmatrix}.$$

ІРЗ другого ЕОК характеризуються матрицями $|C_{\mu 2}|$ і $|\Delta T^R_{\mu 2}|$:

$$C^R_{\mu 2} = \begin{vmatrix} 37\ 46\ 17\ 36\ 47\ 26\ 17\ 45 \\ 30\ 12\ 34\ 29\ 15\ 27\ 28\ 36 \\ 42\ 45\ 13\ 19\ 34\ 35\ 20\ 15 \end{vmatrix}, \quad \Delta T^R_{\mu 2} = \begin{vmatrix} 10\ 10\ 41\ 44\ 14\ 49\ 12\ 48 \\ 14\ 24\ 45\ 13\ 15\ 31\ 33\ 31 \\ 10\ 20\ 40\ 13\ 30\ 13\ 41\ 26 \end{vmatrix}.$$

28 41 29 18 32 16 13 36 26 11 14 27 34 44 32 45

ІРЗ третього ЕОК характеризуються матрицями $|C_{\mu 3}|$ і $|\Delta T^{R}_{\mu 3}|$:

$$C^{R}_{\mu 3} = \begin{vmatrix} 19\ 36\ 41\ 31\ 36\ 24\ 12\ 13 \\ 19\ 30\ 19\ 47\ 31\ 20\ 14\ 20 \\ 44\ 44\ 43\ 25\ 11\ 33\ 17\ 46 \\ 41\ 23\ 16\ 44\ 18\ 20\ 15\ 22 \end{vmatrix}, \Delta T^{R}_{\mu 3} = \begin{vmatrix} 49\ 12\ 13\ 17\ 47\ 21\ 21\ 10 \\ 41\ 18\ 30\ 41\ 30\ 25\ 47\ 12 \\ 22\ 25\ 20\ 32\ 16\ 45\ 25\ 26 \\ 25\ 13\ 13\ 36\ 33\ 36\ 21\ 16 \end{vmatrix}.$$

ІРЗ четвертого ЕОК характеризуються матрицями $|C_{\mu 4}|$ і $|\Delta T^{R}_{\mu 4}|$:

$$C^{R}_{\mu 4} = \begin{vmatrix} 36\ 22\ 29\ 15\ 24\ 41\ 35\ 22 \\ 36\ 30\ 14\ 44\ 31\ 36\ 32\ 48 \\ 42\ 45\ 22\ 16\ 17\ 48\ 36\ 45 \\ 16\ 21\ 16\ 11\ 25\ 44\ 26\ 48 \end{vmatrix}, \Delta T^{R}_{\mu 4} = \begin{vmatrix} 47\ 49\ 44\ 36\ 18\ 14\ 46\ 19 \\ 28\ 33\ 43\ 40\ 46\ 36\ 35\ 49 \\ 29\ 13\ 29\ 24\ 28\ 17\ 27\ 32 \\ 33\ 12\ 30\ 30\ 48\ 41\ 48\ 31 \end{vmatrix}.$$

Параметром каналу зв'язку є час $t_{\mu i}$ пересилання μ–ї задачі і–го ПМ в R–й p–ПМ, що також задається матрицями $|t_{\mu i}|$:

$$t_{\mu i} = \begin{vmatrix} 0\ 0\ 0\ 0\ 0\ 0\ 0 \\ 6\ 5\ 4\ 6\ 9\ 3\ 9\ 3 \\ 9\ 5\ 3\ 5\ 10\ 9\ 1 \\ 7\ 4\ 6\ 3\ 2\ 2\ 1\ 1 \end{vmatrix}, t_{\mu i} = \begin{vmatrix} 0\ 10\ 10\ 5\ 9\ 10\ 8\ 10\ 3 \\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\ 4\ 9\ 5\ 8\ 6\ 10\ 5\ 10\ 7 \\ 6\ 5\ 2\ 8\ 10\ 9\ 1\ 2 \end{vmatrix},$$

$$t_{\mu i} = \begin{vmatrix} 6\ 2\ 2\ 3\ 4\ 1\ 10\ 5\ 2 \\ 10\ 10\ 7\ 7\ 10\ 2\ 3\ 3 \\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 9\ 1 \\ 10\ 2\ 3\ 10\ 6\ 1\ 7\ 5\ 0 \end{vmatrix}, t_{\mu i} = \begin{vmatrix} 2\ 6\ 7\ 8\ 3\ 7\ 1 \\ 8\ 9\ 8\ 2\ 4\ 10\ 9\ 10 \\ 4\ 6\ 1\ 9\ 5\ 7 \\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \end{vmatrix}.$$

Припустимо, що відмова відбулася на першому ЕОК. Ця подія еквівалентна переходу мережі в стан $s_1 = (1, 0, 0, 0)$ і деградації на величину $\delta E = E_0 - E_v = 395 - 292 = 103$. Розв'яжемо задачу перерозподілу ІРЗ у мережі між R працездатними ЕОК.

У результаті розв'язання задачі (3.30)–(3.33) ЦЛП із БЗ отримано вектор d = 00111111111111111111101110111011 і значення функціонала $E_v(d)$ = 366. Отже, перерозподіл ІРЗ забезпечить збільшення функціональної потужності мережі на величину $E_в(d)$ = 366 – 292 = 74 (або на 25,3 %). Таким чином, у стані s_1 у другому ЕОК залишаться на виконанні всі ІРЗ, які були в ньому. У третьому ЕОК будуть виконуватися 1, 2, 3, 4, 5, 7, 8 власні ІРЗ і 3, 7 ІРЗ першого (що відмовив) ЕОК. У четвертому ЕОК будуть розв'зуватися 1, 3, 4, 5, 7, 8 власні ІРЗ і 4, 5, 6, 8 першого ЕОК. Отже, у випадку відмови першого ЕОК після розв'язання задачі (3.111)–(3.113) операторові видаються такі рекомендації:

1. Якщо відкинути ІРЗ першого ЕОК, то функціональна потужність мережі на підпорядкованому рівні управління складе E_v = 292, тобто відбулася функціональна деградація мережі на величину δE = 103.

2. Якщо здійснити перерозподіл ІРЗ у мережі в даному s_v – стані, то можна збільшити $Eв(d)$ на 74 (25,3 %), що призведе до відкидання 1, 2 ІРЗ першого ЕОК, 6 ІРЗ третього ЕОК і 2, 6 ІРЗ четвертого ЕОК.

З погляду ухвалення рішення становить інтерес оцінка величини $Eв(d)$ у випадку переходу мережі зі стану s_0 у будь–який інший стан s_v на основі обраних параметрів ІРЗ і каналів зв'язку, заданих матрицями $|\beta_{\mu i}|$, $|C^R_{\mu 1}|$ і $|\Delta T^R_{\mu l}|$. Результати експериментальної оцінки цієї величини наведені в табл. 3.3. Як видно з табл. 3.3, при однаковій кількості ЕОК, що відмовили, найбільший виграш досягається у випадку відмови ЕОК, на якому розв'язувалися більше значущі ІРЗ, що не суперечить здоровому глузду. З погляду визначення ефективності застосування пропонованого алгоритму становить інтерес також оцінка того, як змінюється величина виграшу при розв'язанні задачі перерозподілу ІРЗ ЕОК, що відмовили, від загальної кількості ЕОК. У моделюванні кількість ЕОК змінювалася від 2 до 10, кожний з яких розв'язував у вихідному стані s_0 однакову кількість ІРЗ – по 4. Результати такого дослідження подані в табл. 3.4, у ній наведені середні значення виграшу $E_в(d)$ за важливістю розв'язуваних у мережі ІРЗ за рахунок їхнього оптимального перерозподілу. Природно, величина цього виграшу $E_в(d)$ залежить від наявності необхідних ресурсів у мережі й припустимого часу передачі ІРЗ р–пм. Тому у випадку перевищення E_v над E_v* (тобто вигідніше відкинути ІРЗ ЕОК, що відмовили, тому що їхній перерозподіл не поліпшить показник функціональної потужності мережі) операторові видаються відповідні рекомендації для ухвалення рішення. При цьому можуть бути уведені додаткові ресурси, що потребуватиме розв'язання задачі перерозподілу ІРЗ у мережі з новими даними.

Таким чином, у ході моделювання показано розв'язання задачі перерозподілу ІРЗ у мережі з урахуванням вимог, що змінюються, до характеристик потоків інформації й важливості ІРЗ. Варто підкреслити універсальність алгоритму, що дозволяє врахувати у вигляді обмежень задачі ЦЛП із БЗ вимоги до характеристик потоків інформації в мережі в умовах виникнення відмов.

Оцінка показників оперативності й вірогідності результатів моделювання

Для оцінки вірогідності й оперативності результатів, одержуваних за допомогою розробленої моделі, скористаємося методикою, запропонованою в роботі [202].

Для порівняння виберемо такі моделі перерозподілу ІРЗ у мережі:

1) імітаційна модель, розроблена в даній роботі;

2) аналітична модель багатокритеріального статичного розподілу ІРЗ у мережі, запропонована в роботі [62];

3) імітаційна модель статичного розподілу ІРЗ у мережі, запропонована в роботі [32].

Таблиця 3.3

Значення показників функціональної потужності й коефіцієнта збереження ефективності після перерозподілу завдань у мережі

s_v	*сост.*	E_0	$E_{рек}$	$E_в$, %	$k_{се}$, %	s_v	*сост.*	E_0	$E_{рек}$	$E_в$, %	$k_{се}$, %
1	0 0 0	292	366	25	20	0	1 1 0	185	287	55	30
0	1 0 0	274	361	30	20	0	1 0 1	192	313	65	30
0	0 1 0	306	361	20	15	0	0 1 1	224	307	40	20
0	0 0 1	31	369	20	15	1	1 1 0	82	173	110	25
1	1 0 0		298	75	30	1	1 0 1	89	239	170	40
1	0 1 0		296	45	25	1	0 1 1	121	202	70	20
1	0 0 1		317	50	30	0	1 1 1	101	202	100	25

Таблиця 3.4

Виграш за рахунок перерозподілу завдань у мережі при різній кількості ПМ, що відмовили

Кількість р-ПМ, %	*Ев*(*d*), %	Кількість р-ПМ, %	*Ев*(*d*), %	Кількість р-ПМ, %	*Ев*(*d*), %
10	16	35	32	60	57
15	18	40	35	65	60
20	22	45	40	70	75
25	25	50	46	75	81
30	28	55	53	80	87

Оцінка показника оперативності кожної моделі визначається ймовірністю своєчасного одержання результатів моделювання (Р). Вірогідність моделювання залежить від важливості й способу обліку основних особливостей функціонування (факторів) мережі й може бути оцінена показником вірогідності (*R*) [202]..

Аналіз результатів розрахунку показує, що показник оперативності істотно залежить від розмірності задачі. При її невеликій розмірності (n<50, m<50) застосування кожної з трьох моделей дозволить вчасно одержати результати розв’язання (показник оперативності Р>0,9). Однак збільшення кількості ІРЗ у мережі і її вузлах призведе до збільшення розмірності задачі. Тоді найкращим для одержання оптимального розв’язку задачі є застосування моделі 1. Аналіз залежності показника оперативності від розмірності задачі вказує на необхідність зменшення часу її

розв’язання, що може бути досягнуто за рахунок реалізації моделей на паралельних обчислювальних структурах.

При оцінці оперативності розв’язання задач управління на мережі цікавим було б оцінити вплив різних сортувань коефіцієнтів у функціоналі й обмеженнях, розглянутих раніше, на цей показник. Результати наведені в табл. 3.5, де як алгоритми для дослідження були обрані:

- точні алгоритми (1 – алгоритм Балаша; 2 – багатоетапний алгоритм);

- наближені алгоритми (1 – алгоритми з тимчасовою складністю $O(n)$; 2 – алгоритми з тимчасовою складністю $O(n^2)$ при сортуванні за обмеженням; 3 – алгоритми з тимчасовою складністю $O(n^2)$ при сортуванні за функціоналом; 4 – алгоритми з тимчасовою складністю $O(n^2)$ при сортуванні за відношенням; 5 – алгоритми з тимчасовою складністю $O(n^3)$; 6 – паралельні алгоритми; 7– адаптивний алгоритм).

Таблиця 3.5

Оцінка показника оперативності

Характеристика	Алгоритм								
	Точний		Наближений						
1	2	3	4	5	6	7	8	9	10
1. НШ 2. СВР (с.) 3. ПО	30 23,86 0,999	30 5.13 1	30 0,0 1	30 0.25 1	30 0,11 1	30 0,27 1	30 0,56 1	30 * 1	30 5,13 1
1. НШ 2. СВР (с.) 3. ПО	50 2100,4 0,082	50 7,56 1	50 0,05 1	50 1,12 1	50 0,75 1	50 1,05 1	50 2,01 1	50 * 1	50 7,26 1
1. НШ 2. СВР (с.) 3. ПО	100 – –	100 9,39 0,999	100 0,14 1	100 3,21 1	100 2,46 1	100 3,17 1	100 7,35 1	100 * 1	100 9,3 1
1. НШ 2. СВР (с.) 3. ПО	150 – –	150 61,94 0,945	150 0,31 1	150 14,17 0,99	150 9,76 0,999	150 13,30 0,999	150 47,9 0,976	150 * 1	150 44,90 0,98

Продовження табл. 3.5

1	2	3	4	5	6	7	8	9	10
1. НШ 2. СВР (с.) 3. ПО	200 – –	200 310,81 0,439	200 0,73 1	200 46,04 0,97	200 32,40 0,996	200 45,53 0,98	200 89,3 0,866	200 * 1	200 45,2 0,98
1. НШ 2. СВР (с.) 3. ПО	250 – –	250 1560,8 0,109	250 1,54 1	250 115,68 0,78	250 91,68 0,859	250 111,8 0,8	250 228,1 0,545	250 * 1	250 1,5 1
1. НШ 2. СВР (с.) 3. ПО	300 – –	300 2700,3 0,064	300 2,1 1	300 171,15 0,650	300 158,3 0,679	300 165,7 0,662	300 310,5 0,439	300 * 1	300 2,4 1

1. НШ	400	400	400	400	400	400	400	400	400
2. СВР (с.)	–	–	2,27	222,9	193,1	211,9	532,9	*	2,5
3. ПО	–	–	1	0,554	0,606	0,572	0,286	1	1

Результати розрахунків показані на рис. 3.76, з якого випливає, що при рівні оперативності $P \geq 0{,}9$ застосування точних алгоритмів можливо при невеликій розмірності розв'язуваної задачі управління – до 250 змінних, а наближених алгоритмів до 400.

Рис. 3.76. Залежність ПЗ від розмірності задачі

Порівняння значення показника вірогідності моделей показує, що аналітична модель, що має середні значення показника оперативності, має досить низьке значення вірогідності внаслідок слабкого обліку найбільш значущих факторів. Невелика різниця між моделями 1 і 3 пояснюється тим, що вони тією або іншою мірою враховують основні фактори, але при цьому їхня спрямованість відрізняється: модель 1 більше пов'язана з дослідженням різних методів динамічного перерозподілу ІРЗ у мережі, а модель 3 – з дослідженням методів передачі інформації з каналів зв'язку в мережі. Однак при цьому модель 1 має більш високий показник Р.

Слід зазначити, що оперативність розв’язання задач реконфігурації БД мережі й відновлення мережі така сама, як і в задачі розподілу завдань у мережі, розглянута в даному пункті.

Оцінка ефективності оптимального планування відновлення мережі

З погляду можливості відновлення несправностей функціональних елементів мереж, що виникають, види відмов умовно можна розділити на відмови, які можуть бути усунуті за допомогою наявних технічних засобів і особового складу, і відмови, які вимагають для відновлення системи додаткових засобів і сил. Відмови обох типів будемо вважати відновлюваними, якщо їх можна усунути за деякий припустимий час $\tau_д$, що визначається задачами, розв'язуваними в цей момент у системі управління. Виявлення покладає на засоби контролю працездатності елементів мережі

(кількісний і допусковий вимір параметрів) і контролю функціонування мережі (перевірка виконання функцій мережі, діагностичний і прогнозуючий контроль). Усунення несправностей планується адміністратором мережі на основі баз даних, що дозволяють будувати сіткові графіки для виконання ремонтних і регламентних робіт на усіх елементах мережі. Приріст коефіцієнта готовності системи за рахунок її відновлення дорівнює $\Delta K_г = K_г(V) - K_г$, де $K_г(V)$ – значення коефіцієнта готовності системи з урахуванням її відновлення; V – імовірність відновлення системи. Позначимо ймовірність відновлення мережі у випадку використання стратегії відновлення a через $V(a_0)$. Під стратегією a потрібно мати на увазі таку стратегію, коли всі наявні засоби й фахівці перекидаються на комплекс робіт, що має на даний час t_i найбільший час виконання, а засоби, що залишилися, розподіляються по інших комплексах за рівномірним законом розподілу, що відповідає найгіршому випадку з погляду розв'язання задачі відновлення на основі моделі ЦЛП із розглянутої раніше. Припустимо, що при здійсненні оптимального планування на основі моделі, розглянутої раніше, у k раз зменшується середній час відновлення системи, тобто в k раз зростає параметр $a_0 = \frac{\tau_д}{T_{во}}$, де $\tau_д$ – припустимий час простою. Неважко показати, що приріст ΔK при збільшенні параметра a_0 описується функцією

$$\Delta K'(\Delta K, \Delta K^*) = K_г \; НШ \; f(a_0, K), \text{ де } f(a_0, k) = e^{-a_0} - e^{-a_0 k} \qquad (3.115)$$

для експонентного закону розподілу;

$$f(a_0, k) = e^{-2a_0} - e^{-2a_0 k} + 2a_0(e^{-2a_0} - ke^{-2a_0 k}) \qquad (3.116)$$

для закону Ерланга. Аналіз залежностей (3.115)–(3.116) показав, що залежно від вихідних значень a_0 можливі максимальні прирости коефіцієнта готовності лежать у діапазоні від 0,1 до 10 % (див. табл. 3.6).

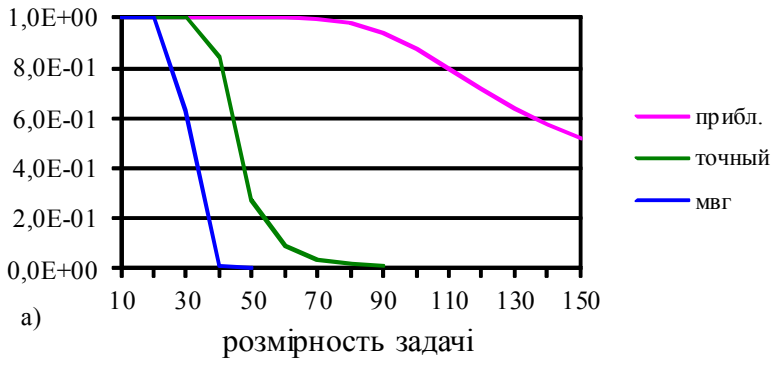
Таблиця 3.6

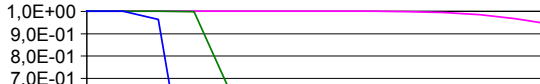
a		0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1,0
$K_Г = 0,8$	k=2	0,0137	0,0237	0,0353	0,038	0,0396	0,0399	0,0395	0,0386	0,037	0,064
	k=3	0,0262	0,043	0,053	0,059	0,0613	0,0613	0,059	0,057	0,054	0,051
$K_Г = 0,9$	k=2	0,008	0,019	0,0172	0,019	0,0221	0,0222	0,0224	0,0222	0,022	0,02
	k=3	0,015	0,024	0,03	0,032	0,035	0,034	0,033	0,032	0,031	0,03
$K_Г = 0.99$	k=2	0,001	0,0014	0,0019	0,0022	0,0023	0,0024	0,0025	0,0024	0,0023	0,002
	k=3	0,002	0,0026	0,0033	0,0036	0,0038	0,0038	0,0037	0,0035	0,0034	0,003
$K_Г = 0,999$	k=2	0,0001	0,0001	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002
	k=3	0,0002	0,0002	0,0003	0,0004	0,0004	0,0004	0,0004	0,0004	0,0003	0,0003
Значення функції $f(a,k)$ для закону Ерланга											
$K_Г = 0,8$	k=2	0,009	0,033	0,042	0,084	0,096	0,097	0,092	0,083	0,073	0,064

	k=3	0,015	0,051	0,085	0,11	0,11	0,1	0,094	0,084	0,074	0,065
$K_Г = 0,9$	k=2	0,005	0,019	0,035	0,047	0,054	0,055	0,051	0,047	0,042	0,037
	k=3	0,009	0,029	0,048	0,059	0,061	0,058	0,053	0,047	0,042	0,037
$K_Г = 0.99$	k=2	0,0005	0,0021	0,0038	0,0052	0,0059	0,006	0,0056	0,005	0,004	0,004
	k=3	0,0009	0,0031	0,0053	0,0065	0,0068	0,0064	0,006	0,005	0,004	0,004
$K_Г = 0,999$	k=2	0,0001	0,0002	0,0004	0,0005	0,0006	0,0006	0,0006	0,0005	0,0004	0,0004
	k=3	0,0001	0,0003	0,0005	0,0007	0,0007	0,0006	0,0006	0,0005	0,0005	0,0004
Значення функції $f(a, k)$ для експоненціального закону											

При експериментальному дослідженні ефективності оптимального планування були отримані дві точки оцінки відношення $k^* = \frac{T_{B0}}{T_B^*}$ при 50 реалізаціях вимірів для кожної точки, де $T_{B\ 0\text{–середній}}$ – час відновлення мережі при використанні стратегії a розподілу робіт, а T^*_B – середній час відновлення мережі при оптимальному плануванні на основі розв'язання задачі (3.104)–(3.106). Дві точки отримані при різній кількості змінних у задачі (3.104)–(3.106). Перше значення K^* = 2,1 отримано при кількості змінних 20, а друге отримано при кількості змінних 30 і кількості обмежень 100. Таким чином, можна зробити висновок, що зі збільшенням кількості відмов роль оптимального планування з погляду управління мережею зростає. Із проведеного раніше аналізу видно, що розглянуті тут задачі розподілу зон управління в мережі, оптимального пошуку інформації в базах даних мережі й оптимального планування передислокації рухомих пунктів управління мережі зводяться до ЗНП.

Оцінка показника оперативності їх розв'язання при різних значеннях розмірності розв'язуваних завдань і допустимості часу розв'язання ТД, дорівнюють відповідно 3, 5 і 10 хвилин (рис. 3.77). Для порівняння на цьому ж рисунку наведене значення показника оперативності при вирішенні цих завдань управління на основі методу гілок і меж.





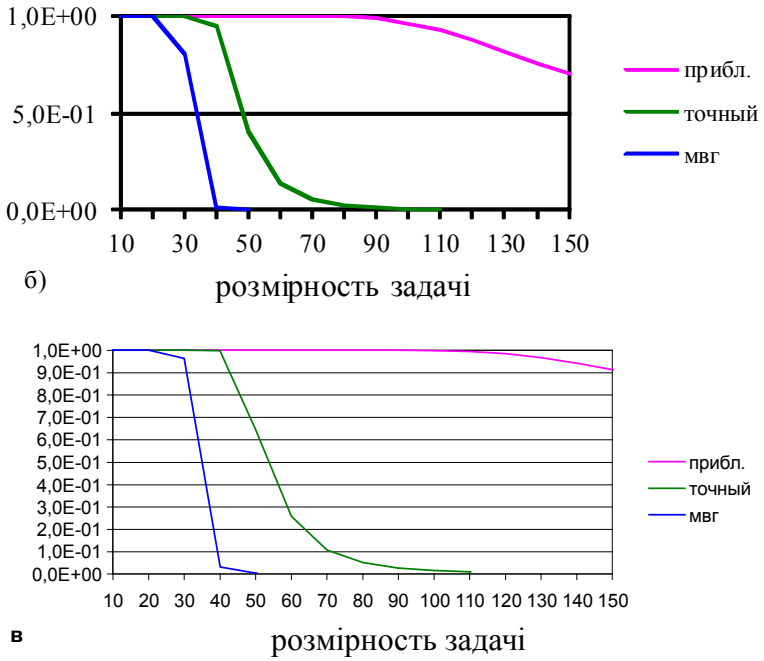


Рис. 3.77. Залежність ОП від розмірності задачі для $T_д$ = 3 хв (а), для $T_д$ = 5 хв (б), $T_д$ = 10 хв (в)

Оцінка показника достовірності моделювання

Оцінка повноти і точності відображення в моделях основних факторів (з урахуванням їх важливості) визначає достовірність результатів, одержуваних з використанням різних моделей. Для проведення такої оцінки приймемо методику, описану в роботі [202]. Значення коефіцієнта важливості факторів у відповідності з методикою [202] можуть бути визначені експертним шляхом. Результати такої експертизи, виконані представниками Харківського військового університету, групи АСУ Головного управління Генерального штабу Збройних Сил України з використанням 100–бальної шкали оцінок, подані в табл. 3.7.

Таблиця 3.7

Вагові характеристики факторів, що враховуються

№	Фактор	Вага фактора	Модель		
			1	2	3
1	2	3	4	5	6
1	Стійкість обчислювальних вузлів мережі до впливу перешкод	0,078	*Н*	*К*	*П*
2	Стійкість КЗ до впливу перешкод	0,073	*Н*	*Н*	*Н*
3	Топологічна структура мережі	0,072	*Н*	*Н*	*Ф*
4	Пропускна здатність ліній зв'язку	0,072	*Н*	*Н*	*Ф*

5	Час відновлення обчислювальних вузлів мережі після відмов	0,065	*П*	*К*	*П*
6	Час відновлення КЗ після відмови	0,065	*П*	*Ф*	*Н*
7	Продуктивність ОВМ в обстановці без перешкод	0,065	*Н*	*К*	*П*
8	Час відновлення обчислювальних вузлів мережі після збою	0,062	*П*	*К*	*К*
9	Інтенсивність вхідного інформаційного потоку	0,058	*Н*	*Ф*	*Ф*
10	Інтенсивність зовнішнього впливу перешкод	0,055	*Н*	*К*	*Н*
11	Потужність зовнішнього впливу перешкод	0,055	*Н*	*К*	*К*
12	Об'єм одночасно розв'язуваних задач на ОВМ ВР	0,045	*Н*	*Ф*	*Н*
13	Алгоритми комутації й маршрутизації в мережі передачі даних	0,040	*К*	*К*	*Н*
14	Прогнозування стану елементів ВР	0,040	*К*	*П*	*П*
15	Імовірність перекручування символу при передачі по КЗ	0,039	*П*	*К*	*Н*
16	Час реакції системи керування обробкою інформацією на виникаючі відмови	0,032	*Н*	*К*	*П*
17	Пріоритетна обробка повідомлень	0,030	*Н*	*К*	*К*
18	Використання методів підвищення вірогідності передачі даних по КЗ	0,027	*П*	*К*	*М*
19	Параметри систем управління обробкою інформації	0,022	*Н*	*П*	*Ф*
20	Розміри буферів для проміжного зберігання повідомлень	0,018	*П*	*К*	*Н*
21	Територіальний розподіл вхідного інформаційного потоку	0,016	*К*	*К*	*К*
22	Метеорологічні умови	0,011	*К*	*Ф*	*К*
23	Забезпечення конфіденційності передачі повідомлень	0,006	*К*	*К*	*К*
24	Цілі операцій функціональних підсистем, що змінюються	0,004	*К*	*К*	*К*
25	Рельєф місцевості	0,001	*К*	*К*	*К*
26	Інші малозначущі фактори (з вагою < 0,001)		*К*	*К*	*К*

Достовірність результатів, одержуваних при моделюванні функціонування відмовостійкості мереж, залежить від важливості і способу обліку кожного фактора в моделях і може бути оцінена показником достовірності [202]

$$R = 1 - \sum_{j=1}^{4} \beta_j \sum_{i \in q_j} \alpha_j . \qquad (3.117)$$

Значення β_j визначає похибку, внесену в розрахунки внаслідок неточного (узагальненого) обліку факторів. При цьому

$\beta_j =$
0 – при безпосередньому обліку факторів (j = 1)(позначення – Н);
0.445 – при простому узагальненні (j = 2) (позначення – П);
0.6 – при функціональному узагальненні (j = 3) (позначення – Ф);
1.33 – при непрямому узагальненні (j = 4)(позначення – К).

Значення показника вірогідності в аналізованих моделях подані в табл. 3.8, звідки випливає, що аналітична модель, що має середні значення показника оперативності, має досить низьке значення вірогідності внаслідок слабкого обліку найбільш значущих факторів. Менша різниця між моделями 1 і 3

пояснюється тим, що вони тією або іншою мірою враховують основні фактори, але при цьому їхня спрямованість відрізняється: модель 1 більше пов'язана з дослідженням різних методів динамічного перерозподілу ІРЗ у мережі, а модель 3 – з дослідженням методів передачі інформації з каналів зв'язку в мережі.

Таблиця 3.8

Значення показника R вірогідності моделювання

Номер моделі	Вірогідність (R)
1	0,7487
2	0,1505
3	0,5451

Вправи

1. Поясніть суть принципу оптимізації по напрямку в n–мірному одиничному кубі й принцип виділення коридору при розв’язанні задачі 0,1– рюкзак.

2. Чому при розв’язанні задач булевого програмування на трикутному графі потрібне сортування коефіцієнтів функціонала в порядку їхнього зменшення з номерів вершин трикутного графа, що є еквівалентом n–мірного одиничного куба?

3. Чому при малих розмірностях оптимізаційної задачі алгоритми на основі ідей методу галузей і границь мають меншу часову складність порівняно з ранговими алгоритмами, а зі збільшенням розмірності задач їхня часова складність різко зростає?

4. Розв’яжіть задачу прикладу 3.1 з використанням стратегій max і min, порівняєте отриманий результат і поясніть, чому ці стратегії дають наближений розв’язок.

5. Як, використовуючи ідеї рангового підходу, можна розв’язати систему цілочисельної лінійних рівнянь із булевими змінними?

6. Як, використовуючи рангові алгоритми для розв’язання ЗНП, можна побудувати алгоритми розв’язання ЗНР?

7. Поясніть, чому зі збільшенням розмірності розв'язуваних задач у рангу в наближених рангових алгоритмах похибка зменшується зі збільшенням розмірності розв'язуваних задач.

РОЗДІЛ 4. Загальна схема розв’язання задач комбінаторної оптимізації й задач нелінійного булевого програмування

4.1. Загальний підхід до розв’язання задач комбінаторної оптимізації й теорії графів

Розглянемо кінцеву множину довільних об'єктів теорії графів або конфігурацій комбінаторної оптимізації.

У загальному випадку об'єкт може бути довільним, але має бути визначена кінцева множина елементів $\Omega = \{\omega_l\}$ або підмножина $L_i \in \Omega$ і правила R, що дозволяють формувати об'єкти з вихідних елементів або підмножин L_i, які належать множині Ω. Нехай задано деяке розбиття множини Ω на сімейства підмножин $\{L_i\}$, що $\bigcup_i L_i = \Omega$, і L_i описують об'єкти, що цікавлять нас, які складаються з таких базових елементів $\{l_i\}$, що $\bigcup_i l_i = \Omega$, і правило R, що дозволяє з базових елементів визначати вагові характеристики довільних об'єднань $L_k \cup L_p \in \Omega$, що характеризує властивості $\{v\}$, і потрібно визначити об'єкт із властивістю, що цікавить нас $v^* \in \{v\}$. Представимо множину всіх можливих об'єднань підмножин L_i у вигляді графа $D_{\equiv}$ (рис. 4.1) з паралельно ярусною структурою, що складається з n горизонтальних ліній з вершинами 1, 2, …, n і n ярусів, кожний з яких містить всі вершини графа $D_{\equiv}$, при цьому кожній вершині графа $D_{\equiv}$ поставимо у відповідність базовий елемент l_i. У графі $D_{\equiv}$ довільна вершина i може бути досягнута шляхами рангів r = 1, r = 2, …, $r = n - 1$, а довільному шляху μ_{st}, що задовольняє правила побудови R і проходить через вершини (j, p, …, k, t), відповідає об'єднання базових елементів $(l_j \cup l_p \cup \ldots \cup l_k \cup l_t)$, визначальний деякий об'єкт $L_i \in \Omega$. Довжина цього шляху $d(\mu_{st})$ визначається за правилами, що належить множині R. Отже, множина всіх шляхів $m_{si}(r)$ у графі $D_{\equiv}$, що задовольняють правила R, визначає область припустимих розв'язків вихідної задачі з виділення об'єкта з властивістю $v^* \in \{v\}$. Як вихідну вершину в графі $D_{\equiv}$ будемо використовувати фіктивну вершину S, що в деяких випадках зручно ототожнювати з нульовим або вихідним станом системи. Це призводить до того, що максимальний ранг шляху в графі $D_{\equiv}$ стає рівним n, а додавання вершини S до базових елементів системи не змінює їхніх властивостей, обумовлених правилами R.

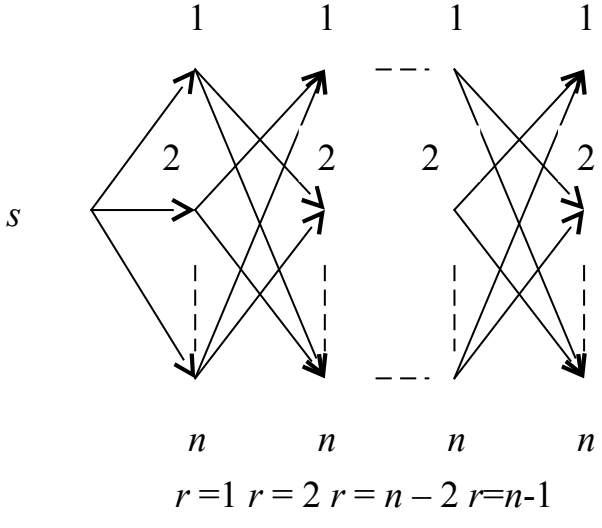


Рис. 4.1. Граф $D_{\equiv}$

Множина складних систем $\{F_i\}$, що мають різні властивості, може бути відображена за допомогою деякої підмножини графів $\{G_i\}$. Розглянемо довільний n - вершинний граф $G(V,E) \in \{G_i\}$, що описує стан системи $F \in \{F_i\}$ з кінцевою кількістю станів n. Вершини $\{i\} \in V$ графа $G(V, E)$ відповідають можливим станам системи, шляхи в графі $G(V, E)$ обумовлені послідовністю проходження вершин $\{v_i\}$ і ребер $\{(i, j)\}$, характеризують можливий порядок досягнення стану $i = p$ з деякого вихідного стану s. Важливою характеристикою шляху є ранг шляху r - кількість ребер, що утворюють шлях. У графі $G(V, E)$ максимальне значення рангу $r = n - 1$, і в загальному випадку ранг довільного шляху μ_{sp} характеризує суму початкового стану, кінцевого стану й кількість станів, що передують, через які може бути досягнутий стан p з деякого вихідного стану s. Тоді множини шляхів $m_{sj}^r ; j = \overline{(1,n)}$ визначають способи досягнення стану j. Як базові елементи $\{l_i\}$ виберемо підмножину $\{v_i\} \in V$ вершин графа $G(V, E)$, тоді об'єктам $\{L_j\}$ буде відповідати вся множина об'єктів Ω, які можна породити на множині V, використовуючи правила R, наприклад кліки в довільних графах, незалежні множини, цикли графів, вектори й матриці, що задають деякий об'єкт у $G(V, E)$. Кожний об'єкт будемо характеризувати $m + 1$ ваговою характеристикою, де m – це деякі другорядні характеристики об'єкта, на які в загальному випадку можуть бути накладені такі обмеження, що вони не повинні перевищувати деяких величин $\{b_i\}$ i = (1, 2, …, m), і є один визначальний показник якості об'єкта, побудований з вихідних елементів або їхніх підмножин об'єкт, що належить множині Ω і має певну властивість v. Правила P визначення вагових характеристик об'єктів природно повинні визначатися відповідно до правил формування самих об'єктів і $P \in R$. Таким чином, шляху μ_{sj}^r в графі $D_{\equiv}$ відповідає об'єкт L_j, що може бути побудований з r базових елементів $\{v_j\}$, включаючи елемент j, на основі правил R, а множини шляхів $m_{sj}^r ; j = \overline{(1,n)}$ визначають множини об'єктів L_j, які можна побудувати з r базових елементів $\{v_j\}$, включаючи елемент j. Уведемо узагальнені процедури: A_0 для формування шляхів у графі $D_{\equiv}$, що дозволяє перераховувати всі об'єкти множини $\{L_j\}$, і $A_0^{/}$ – для визначення об'єктів $\{L_j\}$ із властивістю, що цікавить, при цьому розгляд почнемо з випадку, коли визначальною характеристикою об'єктів $\{L_j\}$ є одна вагова характеристика об'єкта.

Процедура A_0

Крок 1. Формуємо в графі $D_=$ з вершини S у множини шляхів $m_{sj}^{r:=1}$ всі можливі шляхи рангу r = 1 , що задовольняють правило R.

Крок 2. На основі шляхів поточного рангу r формуємо всі можливі шляхи рангу r = r + 1 у підмножинах $m_{sj}^{r:=r+1}$, що задовольняють правило R.

Крок 3. Перевіряємо підмножини $m_{sj}^{r:=r}$ поточного рангу r, чи порожні вони, якщо ні, то переходимо до виконання кроку 2, інакше процедура закінчує роботу, оскільки всі об'єкти $L_j \in \Omega$ перераховані.

Процедура A_0 при переході від довільного рангу r до рангу r + 1 дозволяє будувати з об'єктів L_j, що містять r базових елементів v_j, об'єкти L_j, що містять r + 1 базовий елемент v_j.

Узагальнена процедура $A_0^{/}$

Крок 1. З вершини S будуються шляхи рангу r = 1, що задовольняють правила R, і відповідно до правил R визначаються їхні вагові характеристики.

Крок 2. На основі шляхів поточного рангу r будуються всі можливі шляхи наступного рангу $r = r + 1$, що задовольняють правила R на основі такого рекурентного співвідношення:

$$\mu_{SP}^{r+1} = \min_{d_j(m_{sj}^r \cup (j,p))} (\max)\{m_{sj}^r \cup (j,p)\};\ j = \overline{(1,n)};\ p = \overline{(1,n)};\ j \neq p,$$

де вагові характеристики d_l об'єктів визначаються відповідно до правил $P \in R$.

Крок 3. Перевіряємо підмножини m_{sj}^r поточного рангу r, чи порожні вони, якщо ні, то переходимо до виконання кроку 2, інакше процедура закінчує роботу, і з отриманих локальних екстремумів на всіх рангах вибирається глобальний екстремум.

Залежно від властивостей, що цікавлять, об'єктів і правил R може виникнути необхідність у роботі $\grave{A}_0^{/}$ процедури до досягнення деякого конкретного значення рангу $r = k$, наприклад, якщо об'єктом пошуку є кліка максимальної ваги, потужність якої не перевищує k. Слід також зазначити, що в загальному випадку в підмножинах $m_{sj}^r; j = \overline{(1,n)}$ можлива поява однакових шляхів і дублюючі шляхи в цих підмножинах можна видаляти й, крім того, видаляти їх на ярусах.

Цікавим є з'ясувати, у яких випадках процедура дозволяє одержувати точний або наближений розв'язок задачі визначення об'єкта L_j з необхідною властивістю v, пов'язаного з мінімізацією або максимізацією вагової характеристики об'єкта L_j. Для цього більш докладно проаналізуємо процес переходу від вихідного графа $G(V, E)$ з множиною вершин, що

відповідають базовим елементам, з яких за правилами R у графі $G(V, E)$ можуть бути побудовані підмножини, що цікавлять, об'єкти. Особливістю відображення графа $G(V, E)$ в $D_{\equiv}$ є те, що якщо аналізується конкретний об'єкт L_j в $G(V, E)$, то йому в графі $D_{\equiv}$ відповідає деякий шлях μ_{sj}^{r}, побудований на тій самій множині вершин, що й об'єкт в $G(V, E)$, і вагова характеристика об'єкта L_j в $G(V, E)$ використовується для оцінки довжини шляху μ_{sj}^{r} в $D_{\equiv}$, але при цьому множини шляхів у графах $D_{\equiv}$ і $G(V, E)$ збігаються. Припустимо, що об'єктами в $G(V, E)$ є самі шляхи графа $G(V, E)$, тоді відображення графа $G(V, E)$ є відображенням самого себе й граф $D_{\equiv}$ у цьому випадку, як показано в роботах [9, 27], являє собою стягнуте дерево всіх шляхів даного графа, для прочитання яких на кожній горизонтальній лінії дозволяється бувати тільки один раз. І отже, якщо ми в цьому випадку будемо розв’язувати задачу визначення, наприклад, найкоротших гамільтонових шляхів або найкоротших гамільтонових циклів на основі процедури $A_0^{/}$, то перехід до графа $D_{\equiv}$ тільки впорядкує процес пошуку найкоротших гамільтонових шляхів у графі за рахунок формування на основі рекурентного співвідношення (1). Неважко показати, що в цьому випадку оптимальний розв'язок буде точним, якщо процес роботи процедури безперервний, тобто в процесі її роботи або не виникає ситуацій, коли множини $m_{sj}^{r} = \varnothing$, або якщо така ситуація виникла, то всі шляхи, що ведуть у j у графі $G(V, E)$ рангів $r + 1, \ldots, r = n$, довше, ніж шляхи, отримані в j на ранзі $r - 1$ (доведення справедливості даного твердження є елементарним і тому не наводиться). Однак легко побачити, що якщо множина m_{sj}^{r} виявилася порожньою на деякому ранзі $r = q$, то може виявитися, що у вершину j існує довший, ніж побудований процедурою $\grave{A}_0^{/}$, шлях $\mu_{sj}^{r=q}$, але продовження якого надалі може дозволити одержати коротший гамільтонів шлях. Дана ситуація фактично є каменем спотикання при розв’язанні всіх NP-повних задач, і більш ніж трьохсотрічний досвід показує, що в цьому випадку гарантовано одержати точний розв'язок можна тільки повним перебором або неявним повним перебором на основі методу галузей і границь, що при досить великій розмірності задачі є тим самим. Отже, якщо ми розглядаємо відображення графа $G(V, E)$ в $D_{\equiv}$, і об'єктами L_j в $G(V, E)$ є самі шляхи графа, то перехід до аналізу на графі $D_{\equiv}$ не змінює дану ситуацію, і ми зіштовхуємося з тією самою проблемою, що й розв’язання даної задачі на графі $G(V, E)$ без використання перетворень. Однак ситуація в деяких випадках змінюється, з погляду одержання оптимального розв'язку, якщо аналізований об'єкт у графі $G(V, E)$ відмінний від шляху в даному графі, наприклад цикл, кліка й т. д., а множини шляхів в обох графах, як випливає з побудови графа $D_{\equiv}$,

при цьому збігаються. У ведемо процедуру В перетворення об'єктів$\{L_j\}$ у графі $D_=$.

Процедура В

Крок 1. Використовуючи правило R, формуємо множину всіх об'єктів $L_j^{r=2}$, що складаються з $r = 2$ базових елементів з вагами $d_j^{r=2}$, і виділяємо об'єкт L_j^{*r} із мінімальною вагою $d_j^{*r=2}$, якщо таких мало, то виділяємо їх усі.

Крок 2. На основі виділеного об'єкта L_j^{*r} й правила *R* формуємо всі можливі об'єкти $L_j^{r:=r+1}$, що складаються з $r = r + 1$ базових елементів з вагами $d_j^{r:=r+1}$, і виділяємо об'єкт $L_j^{*r:=r+1}$ із мінімальною вагою $d_j^{*r:=r+1}$, якщо таких мало, то виділяємо їх усі.

Крок 3. Перевіряємо на основі L_j^{*r} й правила R, чи можна побудувати об'єкти з $r = r + 1$ базових елементів, якщо ні, то процедура закінчує роботу, інакше переходимо до виконання кроку 2.

У результаті роботи процедури *B* одержимо об'єкти $L_j^{*r=2}$ $L_j^{*r=3}$ … $L_j^{*r=\hat{e}}$, що містять відповідно по 2, по 3 …, k базових елементів. Неважко побачити, що справедливо таке твердження.

Твердження 4.1. Якщо об'єкти $\{L_j\}$ задовольняють властивість *B*, що полягає в тім, що застосування процедури *B* до множини $\{L_j\}$ дозволяє побудувати об'єкти $L_j^{*r=2}$,$L_j^{*r=3}$,…,$L_j^{*r=\hat{e}}$, що містять відповідно по 2, 3 …, *k* базових елементів з мінімальними вагами d_j^{*r1}, то процедура $\grave{A}_0^{/}$ дає точний розв'язок задачі. Твердження є справедливим, оскільки якщо припустити, що властивість *B* виконується й на основі процедури $\grave{A}_0^{/}$ отриманий глобальний екстремум, який не є точним розв'язком, то це можливо, якщо існують об'єкти $L_j^{**r=2}$,$L_j^{**r=3}$,…,$L_j^{**r=\hat{e}}$, довжини яких менше, ніж в об'єктів $L_j^{*r=2}$,$L_j^{*r=3}$,…,$L_j^{*r=\hat{e}}$, що суперечить первісному припущенню, про те, що властивість В виконується. Таким чином, якщо властивість *B* виконується, запропонована процедура $\grave{A}_0^{/}$ буде давати точний розв'язок в умовах безперервності роботи процедури, а якщо не виконується, то наближений. Слід також зазначити, що у випадку виконання властивості *B* наявність порожніх множин m_{sj}^r при роботі процедури $\grave{A}_0^{/}$ означає, що на основі правил *R* побудувати об'єкт $L_j^{r:=r}$, що містить базовий елемент *j*, або неможливо в принципі (останнє потрібно обґрунтовувати для кожної задачі окремо), або будувати даний елемент не має сенсу, оскільки його вагова характеристика буде істотно гірше за ті, що побудовано на ярусі.

Твердження 4.2. Якщо аналізовані об'єкти $\{L_j\}$ у графі $G(V, E)$ відмінні від шляху в даному графі, наприклад цикл, кліка й т. д., то вони завжди задовольняють властивість B графі $D_{\equiv}$.

Доведення. Нехай на основі процедури B у графі $D_{\equiv}$ побудовані всі об'єкти, що містять 2 базових елементи, що задовольняють правило R, виділимо серед них об'єкт $L_j^{*r:=2}$, що має мінімальну довжину $d_j^{*r:=2}$, і побудуємо на його основі всі можливі об'єкти $L_j^{r:=3}$, що містять 3 базових елементи, і виділимо серед них об'єкт $L_j^{*r:=3}$, що має мінімальну довжину $d_j^{*r:=3}$ відносно даного об'єкта. Можна стверджувати, що він є об'єктом з мінімальною довжиною, який можна побудувати, якщо процес побудови безперервний. Дійсно припустимо, що в $G(V, E)$ існує об'єкт $L_j^{**r:=3}$, що має меншу довжину, ніж $L_j^{*r:=3}$, але це можливо, якщо в $G(V, E)$ і відповідно в $D_{\equiv}$ існує об'єкт $L_j^{**r:=2}$ із довжиною, меншою, ніж довжина $L_j^{*r:=2}$, але це суперечить установленому факту, що $L_j^{*r:=2}$ - об'єкт мінімальної довжини в $G(V, E)$, а отже, наше припущення не є правильним, аналогічні міркування можна на основі принципу повної математичної індукції провести і для об'єктів $L_j^{*r:=3} \dots L_j^{*r:=\hat{e}}$, і довільного об'єкта L_j^{*r} і, отже, властивість B для них виконується.

Таким чином, із тверджень 4.1 і 4.2 випливає, що умова того, що відображення G(V, E) у граф $D_{\equiv}$ не є відображенням $G(V, E)$ самого себе, тобто аналізовані об'єкти відмінні від шляху в даному графі $G(V, E)$, є необхідною, але не достатньою умовою для виконання властивості B (достатність, на жаль, необхідно обґрунтовувати для кожної задачі окремо). Отже, для випадку, коли об'єкти $\{L_j\}$ характеризуються однією ваговою характеристикою, можна спростити процедуру $A_0^{/}$ за рахунок виділення на ярусах глобальних екстремумів і наступного формування всіх можливих шляхів наступного рангу в графі $D_{\equiv}$. Позначимо цю процедуру $A_0^{//}$, що буде описано далі.

Процедура $A_0^{//}$

Крок 1. З вершини S будуються шляхи рангу $r = 1$, що задовольняють правила R, і відповідно до правил R визначаються їхні вагові характеристики.

Крок 2. На основі шляхів поточного рангу r будуються всі можливі шляхи наступного рангу $r = r + 1$, що задовольняють правила R, на основі такого рекурентного співвідношення:

$$\mu_{SP}^{r:=r+1} = \min_{d_j(m_{sj}^r \cup (j,p))} (\max)\{m_{sj}^r \cup (j,p)\}; j = \overline{(1,n)}; p = \overline{(1,n)}; j \neq p,$$

де вагові характеристики d_l об'єктів визначаються відповідно до правил $P \in R$, серед них на ярусі виділяється шлях μ_{sj}^{*r}, максимальний або мінімальний на ярусі по довжині, і на його основі формуємо будь-які шляхи рангу $r = r + 1$ у графі $D_{\equiv}$.

Крок 3. Перевіряємо підмножини m_{sj}^{r} поточного рангу r, чи порожні вони, якщо ні, то переходимо до виконання кроку 2, інакше процедура закінчує роботу, і з отриманих на останньому ранзі шляхів вибирається кращий (максимальний або мінімальний) по довжині шлях.

Розглянемо застосування процедури $\grave{A}_0^{/}$ й $\grave{A}_0^{//}$ для розв'язання задач визначення незалежної максимальної множини й мінімального гамільтонового циклу в довільних графах. Остання задача в теорії графів широко відома як задача про комівояжера. Нехай задано граф G (рис. 4.2), зважений по вершинах (ваги вершин наведені в табл. 4.1), і потрібно визначити незалежну максимальну множину вершин.

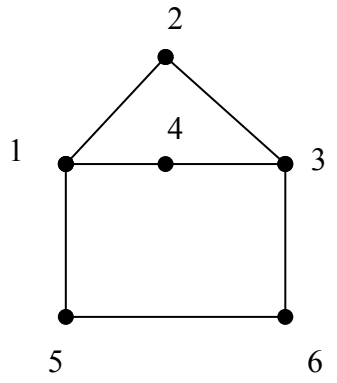


Рис. 4.2. Граф G

Таблиця 4.1

Ваги вершин графа

Номер вершини	1	2	3	4	5	6
Вага вершини	4	1	5	2	2	1

Тут правило R буде полягати в тім, що в множини поєднуваних вершин у графі $D_{\equiv}$ допускається в тому випадку, якщо вони в графі G не зв'язані, а ваговою характеристикою буде сума ваг вершин, об'єднаних у шлях у графі $D_{\equiv}$. Послідовність розв'язання задачі визначення незалежної максимальної множини вершин мінімальної ваги на основі процедури $\grave{A}_0^{/}$ показана в табл. 4.2.

Таблиця 4.2

Процес розв'язання задачі визначення незалежної

максимальної множини

S1(4) 1	___ 1	___ 1	____ 1
S2(1) 2	S62(2) 2	___ 2	____ 2
S3(5) 3	___ 3	___ 3	____ 3
S4(2) 4	___ 4	S624(4) 4	____ 4
S5(2) 5	___ 5	___ 5	____ 5
S6(1) 6	___ 6	___ 6	____ 6

Як випливає з табл. 4.2, оптимальному розв'язку задачі відповідають шляхи S624(4) і S625(4), тобто підмножини вершин {2, 4, 6} і {2, 5, 6} утворять у графі G (рис. 3.6) незалежні максимальні множини з мінімальною вагою 4.

У випадку застосування процедури $\grave{A}_0^{//}$ процес розв'язання задачі може бути поданий у вигляді табл. 4.3.

Таблиця 4.3

Процес розв'язання задачі на основі процедури $A_0^{//}$

S1(4) 1	S31(1) S61(5) 1	_____ 1	_____ 1
S2(1) 2	S42(3) S52(3) S62(2) 2	S542(5) S462(4) 2	 _____ 2
S3(5) 3	S13(3) S53(7) 3	_____ 3	_____ 3
S4(2) 4	S24(3) S54(4) 4	S624(4) S254(5) 4	_____ 4
S5(2) 5	S25(3) S35(7) S45(4) 5	S625(4) S245(5) 5	_____ 5
S6(1) 6	S16(5) S26(2) S46(3) 6	S316(10) S426(4) 6	 _____ 6

Розглянемо розв'язання задачі про комівояжера на основі процедури $\grave{A}_0^{/}$ для графа G повного графа G із кількістю вершин n = 5, заданого матрицею C:

$$C = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ 1 & 0 & 3 & 1 & 4 & 2 \\ 2 & 3 & 0 & 2 & 1 & 1 \\ 3 & 1 & 2 & 0 & 2 & 3 \\ 4 & 4 & 1 & 2 & 0 & 4 \\ 5 & 2 & 1 & 3 & 4 & 0 \end{array}.$$

Процес розв’язання даної задачі процедурою $\grave{A}_0^{/}$ наведений у табл. 4.4.

Таблиця 4.4

Процес розв’язання задачі про комівояжера процедурою $A_0^{/}$

S1(0) 1	21(3) 31(1) 41(4) 51(2) 1	 _____ 1	2134(6) 1	 ____ 1
S2(0) 2	12(3) 32(2) 42(1) 52(1) 2	$(234)^{/}$ (5) 2	 _____ 2	 ____ 2
S3(0) 3	13(1) 23(2) 43(2) 53(3) 3	 _____ 3	 _____ 3	 ____ 3
S4(0) 4	14(4) 24(1) 34(2) 54(4) 4	 _____ 4	 _____ 4	 ____ 4
S5(0) 5	15(2) 25(1) 35(3) 45(4) 5	 _____ 5	 _____ 5	25134(7) 5

У загальному випадку, додаючи будь-яку вершину p до циклу, що містить r вершин, необхідно перевіряти всі r варіантів розміщення вершини p у циклі й серед них вибирати найкоротший. У табл. 4.4 цикл із трьох вершин 234 довжиною 5 є найкоротшим, на його основі можна побудувати цикли, додаючи вершини 1 і 5.

Як видно з рис. 4.3 і 4.4, на наступному ранзі ми можемо сформувати цикли 2134 довжиною 6 і 2534 довжиною 7. Оскільки цикл 2134 найкоротший на ярусі, то його тільки й залишаємо, і на його основі на наступному ярусі формуємо найкоротший цикл із п'яти вершин 25134 довжиною 7. Оскільки кожний цикл доводиться перевіряти на можливість розширення *r* раз, а максимальна кількість шляхів в одній підмножині не може перевищити (n – 1), кількість ярусів дорівнює n, максимальний ранг шляху теж не перевищує n, то загальна складність алгоритму розв’язання задачі про комівояжера не перевищить $O(n^4)$. У загальному випадку при довільних об'єктах складність процедури $\grave{A}_0^{/}$ не перевищить $O(kn^3)$, де k – кількість операцій, необхідних для визначення оптимальної ваги об'єкта при додаванні в нього ще одного базового елемента. Тепер розглянемо

можливість розв’язання задач дискретної оптимізації для випадку, коли об'єкт характеризується m + 1 вагою й на m ваг накладається обмеження. У цьому випадку ми приходимо до задач лінійного й нелінійного булевого програмування. Задачі булевого лінійного й нелінійного програмування є моделями широкого класу прикладних задач у теорії побудови складних систем, і при цьому задачі булевого лінійного програмування належать до класу NP-повних, важко розв'язуваних, задач [162], а ефективні методи розв’язання задач нелінійного булевого програмування з довільними нелінійностями практично відсутні. На сьогодні для кожного типу задачі розробляється свій метод розв’язання. Покажемо, що на основі запропонованої узагальненої процедури може бути розв’язана будь-яка задача булевого програмування.

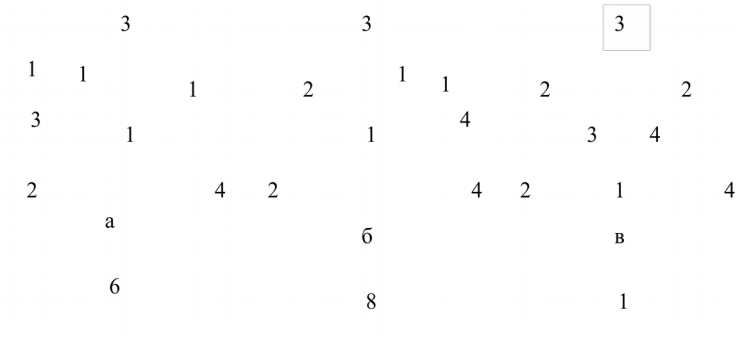


Рис. 4.3. Варіанти розміщення вершини 1 (додаючи вершину 1, ми одержимо цикли (див. рис. 3.6 і 3.7))

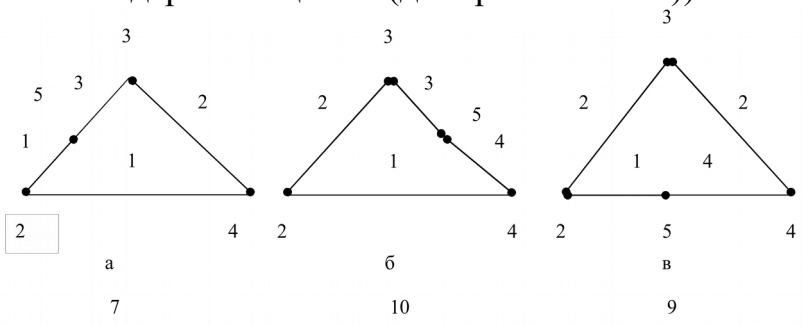


Рис. 4.4. Варіанти розміщення вершини 5

4.2. Загальний метод розв’язання довільних задач булевого програмування

Задачі булевого лінійного й нелінійного програмування є моделями широкого класу прикладних задач у теорії побудови складних систем, і при цьому задачі булевого лінійного програмування належать до класу NP-повних, важко розв'язуваних, задач [78, 79, 162], а ефективні методи розв’язання задач нелінійного булевого програмування з довільними нелінійностями практично відсутні. На сьогодні для кожного типу задачі розробляється свій метод розв’язання. Актуальним є вироблення єдиного

підходу до розв'язання даного класу задач, що забезпечує їхнє розв'язання з необхідною оперативністю й точністю. Розглянемо підхід, що дозволяє розв'язувати задачі булевого лінійного програмування й нелінійного програмування з довільними нелінійностями тим самим алгоритмом, заснованим на ідеях рангового підходу [106].

Формалізація й постановка задачі

Для опису всієї множини адитивних цілочисельних функцій з довільними нелінійностями, обумовленими множиною змінних $\{X_1, X_2, ..., X_n\}$, уведемо поняття «позначка» функції $F(x)$:

$$F(x) = \sum_{j=1}^{p_1} C_{1j} S_1(C_n^1) + \sum_{j=1}^{p_2} C_{2j} S_2(C_n^2) + ... + \\ + \sum_{j=1}^{p_k} C_{kj} S_k(C_n^k) + ... + \sum_{j=1}^{p_n} C_{nj} S_n(C_n^n), \tag{4.1}$$

де $Sr(C_n^r) = S_1 + S_2 + ... + S_{p_r}$ – сума всіх можливих поєднань добутків змінних, що містять у кожному добутку $Sr = X_p X_к ... X_m$ (нелінійності) r різних змінних;

$$p_r = \frac{n!}{r!(n-r)!};$$

C_{rj} – цілочисельні коефіцієнти варті в добутках Sr, що містять r змінних.

Позначимо через H множину всіх функцій, яку можна породити на основі F(x), припускаючи рівними нулю різні поєднання C_{lj} у виразі (4.1). Множина H є повною в тому розумінні, що містить у собі всі можливі нелінійності, що складаються з усіх можливих поєднань змінних, що утворюють ці нелінійності, які можна взагалі побудувати на основі даної підмножини змінних $\{X_1, X_2, ..., X_n\}$.

Неважко показати, що потужність даної множини дуже велика, але кінцева й дорівнює 2^{p_Σ}, де

$$p_\Sigma = 1 + \frac{1}{2}\left[\frac{n!}{1!(n-1)!}\left(\frac{n!}{1!(n-1)!}+1\right) + \frac{n!}{2!(n-2)!}\left(\frac{n!}{2!(n-2)!}+1\right) + ... + \right. \\ \left. + \frac{n!}{k!(n-k)!}\left(\frac{n!}{k!(n-k)!}+1\right) + ... + \frac{n!}{(n-1)!1!}\left(\frac{n!}{(n-1)!1!}+1\right)\right].$$

Слід зазначити, що за допомогою співвідношення (3.29) може бути визначено клас задач дискретної оптимізації, у яких розв'язок визначається тільки поєднанням змінних і не залежить від перестановки змінних в $S_r(C_n^r)$, тобто значення C_{lj} в цих задачах залежить тільки від поєднання змінних в $S_r(C_n^r)$. У загальному випадку задачу булевого програмування можна подати як

$$f(X_1, X_2, ..., X_n) \Rightarrow \max;$$
$$g_j(X_1, X_2, ..., X_n) \le b_j; \quad j = \overline{(1,m)}, \tag{4.2}$$

де $f(X_1, X_2, ..., X_n) \in H$,
$g_j(X_1, X_2, ..., X_n) \in H$;
$b_j \in Z$; Z - *множина цілих чисел*;
$X_i \in [0,1]$.

Розглянемо граф $G(X, E)$ (рис. 4.5), у якому вершини X_i і X_j з'єднані ребром (i, j), якщо вони можуть бути об'єднані в кліку. У графі $G(X, E)$ кожній вершині X_i відповідає змінна X_i.

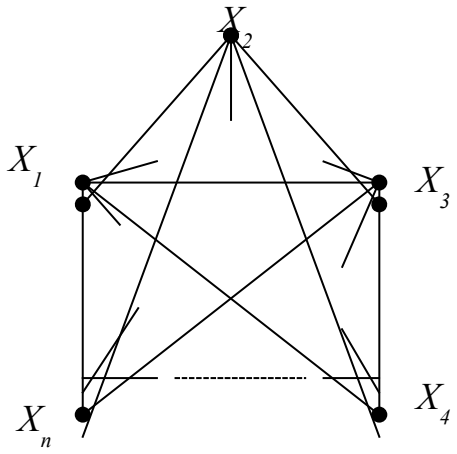


Рис. 4.5. Граф G

Виділимо в графі G довільну кліку $Q = X_p X_r ... X_m$, що складається з r вершин, де $r < n$, і розглянемо її перетин з $S_r(C_n^r) \in f(X_1, X_2, ..., X_n)$, а також з $S_r(C_n^r) \in g_j(X_1, X_2, ..., X_n)$.

Кожен перетин можна характеризувати сумами коефіцієнтів C_{lj}, що стоять при $S_r(C_n^r)$ у функціоналі $f(X_1, X_2, ..., X_n)$ і обмеженнях $g_j(X_1, X_2, ..., X_n)$, при цьому в загальному випадку довільна кліка Q завжди буде характеризуватися відповідною вагою по функціоналу $f(X_1, X_2, ..., X_n)$ і не більш ніж m вагами по обмеженнях $g_j(X_1, X_2, ..., X_n)$. Таким чином, довільна задача булевого програмування може розглядатися як задача знаходження кліки Q^* максимальної ваги по вагах функціонала у графі G, у якій всі m ваг по вагах обмежень не перевищують відповідно b_j. Так, наприклад, якщо розв'язується задача лінійного програмування

$$f(x) = C_1X_1 + C_2X_2 + C_3X_3 + C_4X_4 \to \max;$$
$$B_1X_1 + B_2X_2 + B_3X_3 + B_4X_4 \le b_1; \tag{4.3}$$
$$K_1X_1 + K_2X_2 + K_3X_3 + K_4X_4 \le b_2,$$

то граф G буде мати вигляд як на рис. 4.6.

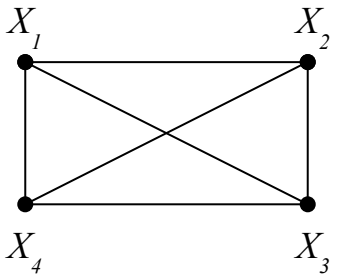


Рис. 4.6. Граф G для задавання лінійного програмування

Як видно з табл. 4.5, кожна вершина графа G характеризується трьома вагами (C_i, B_i, K_i) і для ров'язання задачі в графі G (рис. 4.6) варто знайти таку кліку Q^* зі зважених вершин, щоб її сумарна вага по вагах $\{C_i\}$ була максимальною, і при цьому сумарні ваги по вагах $\{B_i\}$ і $\{K_i\}$ не перевищували відповідно величин b_1 і b_2. Слід зазначити, що в задачах лінійного програмування й з нелінійністю, вище від другої (тобто за наявності у функції мети або обмежень добутків, що складаються з кількості змінних, більше двох), вагові характеристики ребер графа G покладаються рівними нулю. У випадку розв'язання задач квадратичного програмування зручно вводити й вагові характеристики ребер. Так, розглянемо задачу квадратичного програмування вигляду

$$\begin{gathered} f(x) = C_1X_1 + C_2X_2 + C_3X_3 + C_4X_4 + \\ + C_{12}X_1X_2 + C_{13}X_1X_3 + C_{34}X_1X_4 + \\ + C_{23}X_2X_3 + C_{24}X_2X_4 + C_{34}X_3X_4 \to \max, \\ B_1X_1 + B_2X_2 + B_3X_3 + B_4X_4 \le b_1, \\ K_1X_1 + K_2X_2 + K_3X_3 + K_4X_4 \le b_2. \end{gathered} \tag{4.4}$$

Таблиця 4.5

Ваги вершин графа

X_1	C_1	B_1	K_1
X_2	C_2	B_2	K_2
X_3	C_3	B_3	K_3
X_4	C_4	B_4	K_4

Для неї можна поставити у відповідність граф на рис. 4.7.

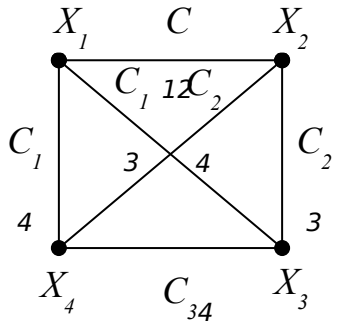


Рис. 4.7. Граф *G* для задавання квадратичного програмування

Як видно з рис. 4.7 і табл. 4.6, вершинам графа, як і в задачі лінійного програмування, відповідають ваги (C_i, B_i, K_i), а ребрам – ваги $\{C_{ij}\}$. Для розв'язання даної задачі в графі *G* потрібно знайти кліку Q^* максимальної сумарної ваги по C_i і C_{ij}, щоб і при цьому сумарні ваги по вагах $\{B_i\}$ і $\{K_i\}$ не перевищували відповідно величин b_1 і b_2. У цьому випадку граф *G* є зваженим і по вершинах, і по ребрах, аналогічно зваженими є й усе кліки даного графа. Варто мати на увазі, що у квадратичній задачі $C_{ij} = C_{ji}$. Таким чином, для розв'язання довільної задачі булевого програмування необхідно побудувати алгоритм, що дозволяє знаходити в заданому графі G, зваженому по вершинах або по вершинах і по ребрах, кліку Q^* максимальної сумарної ваги по вагах функціонала $f(X_1, X_2, ..., X_n)$, у якій всі *m* ваг по вагах обмежень $g_j(X_1, X_2, ..., X_n)$ не перевищують відповідних b_j.

Таблиця 4.6

Ваги вершин і ребер графа

X_1	C_1	B_1	K_1
X_2	C_2	B_2	K_2
X_3	C_3	B_3	K_3
X_4	C_4	B_4	K_4

Розв'язання задачі. Для розв'язання задачі скористаємося поданням вихідного графа *G* у вигляді симетричного дерева шляхів, запропонованого в роботах [6–8]. Сенс такого подання полягає в наступному. Нехай всі можливі стани деякої системи визначаються графом *G*(*V, E*) з n вершинами, де вершини відповідають можливим станам системи. Перейдемо до простору з $(n - 1)^2$ станами. Для цього кожному з *n* станів поставимо у відповідність ще $(n - 1)$ стан, що характеризує спосіб досягнення стану з множини $\{1, 2,..., n\}$. При цьому як додаткові стани визначимо ранг шляху в графі *G*(*V, E*). Тобто з вершини *s* графа *G*(*V, E*) у довільну вершину *j* можна потрапити шляхом рангу $r = 1$, використовуючи одне ребро, шляхом рангу $r = 2$, використовуючи 2 ребра, і т. д., шляхом рангу $r = n - 1$, використовуючи n – 1 ребро. Такий простір станів можна подати у вигляді стягнутого дерева шляхів *D*, графічно воно може бути зображене в такий спосіб, як на рис. 4.8.

Дерево всіх шляхів D містить (n - 1) горизонтальну лінійку і (n - 1) ярус. Для прочитання шляхів на кожній горизонтальній лінійці можна бувати тільки один раз. Виходячи зі стягнутого дерева шляхів, для довільної вершини j множину шляхів, що ведуть у цю вершину з деякої вершини s, можна подати як

$$m_s(j) = m_{sj}^{r=1} \cup m_{sj}^{r=2} \cup .... \cup m_{sj}^{r=n-1};\ j = \overline{(1, n-1)}, \qquad (4.5)$$

де $m_{sj}^r = \{\mu_{sj}^r\}$ – підмножини шляхів з довільної вершини s у деяку вершину j графа $G(V, E)$ рангу r.

Слід зазначити, що дерево всіх шляхів D може будуватися й від конкретної вершини і графа, у цьому випадку вершина $s = i$ і i-та горизонтальна лінійка виключається в D. Наприклад, при $i = 2$ дерево D буде мати вигляд, як на рис. 4.8. Надалі стягнуте дерево шляхів, наведене на рис. 4.8, буде використовуватися для побудови однопрохідних алгоритмів розв'язання задачі (4.2), а дерево D на рис. 4.9 – для побудови n-прохідних алгоритмів, суть побудови яких буде пояснена нижче.

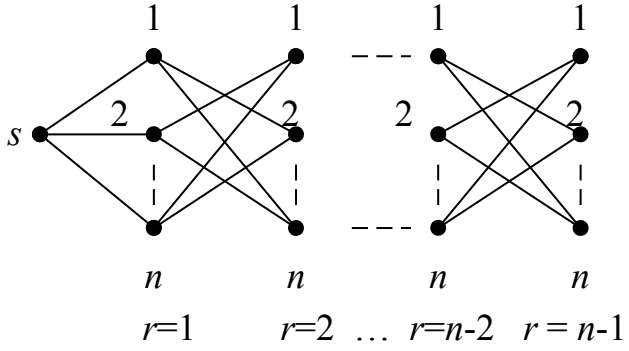


Рис. 4.8. Стягнуте дерево всіх шляхів D графа $G(V, E)$

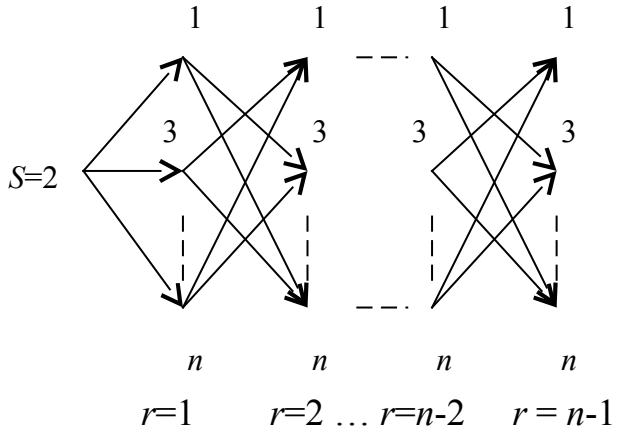


Рис. 4.9. Стягнуте дерево всіх шляхів D графа від вершини s=2 $G(V,E)$

Таким чином, використовуючи граф D і ввівши правила формування шляхів наступного рангу, ми можемо з довільної вершини s поетапно будувати шляхи $\{\mu_{sj}^{r}\}$ довільного рангу аж до рангу $r = n - 1$. У нашій задачі під станом системи ми будемо мати на увазі різні способи об'єднання вершин графа D у кліці. Тоді кожному шляху $\{\mu_{sj}^{r}\}$ рангу r у графі D, що проходить через вершини (v_h, v_k, …, v_p), у вихідному графі G розв'язуваної задачі відповідає кліка з $r - 1$ вершини ($X_h X_k ... Xp$), що характеризується відповідною вагою по функціоналу $f(X_1, X_2,..., X_n)$ і не більш ніж m вагами по обмеженнях $g_j(X_1, X_2,..., X_n)$. Вагові характеристики $\{d_{sj}^{r-1}\}$ довільної кліки $Q^{\text{r-1}}(j)$, що складається з r – 1 вершини й обумовлена одним зі шляхів $\mu_{sj}^{r} \in m_{sj}^{r}$ рангу r у графі D, обчислюються по вагах функціонала шляхом підсумовування коефіцієнтів підмножини $L_f=\{C_{rj}\}$, що стоять при $S_{r-1}(C_n^{r-1}) \in P_f$, де P_f – всі підмножини $\{S_{r-1}(C_n^{r-1})\}_f$, що задовольняють умову $S_{r-1}(C_n^{r-1})_f \cap Q^{\text{r-1}}(j) \neq \varnothing$, а $S_{r-1}(C_n^{r-1})_f$ визначається функціоналом $f(X_1, X_2,..., X_n)$. Аналогічно визначаються вагові характеристики по вагах обмежень шляхом підсумовування коефіцієнтів підмножини $L_B=\{C_{rj}\}$, що стоять при $S_{r-1}(C_n^{r-1}) \in P_B$, де P_B – всі підмножини $\{S_{r-1}(C_n^{r-1})\}_{\text{B}}$, що задовольняють умову $S_{r-1}(C_n^{r-1})_B \cap Q^{\text{r-1}}(\text{j}) \neq \varnothing$, а $S_{r-1}(C_n^{r-1})_B$ визначається обмеженнями $g_j(X_1, X_2,..., X_n)$; $j = \overline{(1, m)}$. Таким чином, вагові характеристики клік $Q^{\text{r-1}}(j)$ шляхів, що m_{sj}^{r} характеризуються множиною, по вагах функціонала й обмежень визначаються відповідно рівностями

$$d_{sj}^{f(\,r-1)} = \sum_{C_{rj} \in L_f} C_{rj} \; ; \; d_{sj}^{B(r-1)} = \sum_{C_{rj} \in L_B} C_{rj} \; . \qquad (4.6)$$

Отже, у графі D кожний шлях має в загальному випадку m + 1 довжину, одну по вагах функціонала й m по вагах обмежень, і для розв'язання поставленої задачі нам у графі D потрібно побудувати шлях максимальної довжини по вагах функціонала від вершин 1, 2,…, n до всіх інших вершин графа й при цьому його довжини по вагах обмежень не повинні перевищувати відповідної величини b_j. Якщо на основі підмножин шляхів $m_{sj}^{r=1}$ у графі D будувати підмножини $m_{sj}^{r=2}$ й так далі до $m_{sj}^{r=n-1}$, то ми змушені будемо побудувати $(n - 1)!$ шляхів, тому для формування шляхів уводиться процедура A, що дозволяє відсікати безперспективні шляхи. Для відсікання безперспективних варіантів у процедурі А пропонується використовувати принцип оптимізації в напрямку до довільної вершини p при формуванні шляхів наступного рангу $m_{sj}^{r=1}$ на основі шляхів попереднього рангу m_{sj}^{r}, запропонованих у роботах [104–107], що для розглянутої задачі визначається таким рекурентним співвідношенням:

$$\mu_{sp}^{r+1} = \max_{j}\{\{\mu_{sj}^{r}\} \cup (j,p)\}; j = \overline{(1,n)}; p = \overline{(1,n)}; j \neq p \,, \qquad (4.7)$$

де (j, p) – ребро графа D;

n – кількість різних вершин у графі D.

Розглянемо можливість побудови n-прохідних і однопрохідних процедур розв'язання задачі (3.30) відповідно на стягнутих деревах, наведених на рис. 4.8 і 4.9.

Перед початком роботи процедури A_1 змінна $i = 1$.

Процедура A_1 з n проходами

Крок 1. Змінна $s = i$ і з вершини s будуються всі можливі шляхи рангу r = 1 до всіх вершин графа D (рис. 4.9), що задовольняють обмеження $g_j(X_1, X_2,..., X_n)$; $j = \overline{(1,m)}$, при цьому довжини по вагах функціонала й обмежень обчислюються відповідно до співвідношень (4.6).

Крок 2. Використовуючи шляхи поточного рангу r, будуються всі можливі шляхи рангу $r = r + 1$, що задовольняють обмеження $g_j(X_1, X_2,..., X_n)$; $j = \overline{(1,m)}$ з використанням рекурентних співвідношень (4.6). При цьому перевірка обмежень і вибір шляху, максимального по вагах функціонала, здійснюється на основі обчислень довжин шляхів по вагах функціонала й обмежень відповідно до співвідношень (4.6). (Слід зазначити, що якщо в процесі застосування рекурентного співвідношення (4.7) виникають кілька шляхів однакової довжини, то необхідно їх усі продовжувати на наступному ранзі.)

Крок 3. Перевіряємо $m_{sj}^{r+1} = \varnothing$, якщо так, то шлях μ_{sj}^{*r} максимальної довжини, отриманий на ранзі r, є локальним екстремумом розв'язуваної задачі, інакше виконуємо наступний крок.

Крок 4. Перевіряємо $i = n - 1$, якщо ні, то $i = i + 1$ і переходимо до виконання кроку 1, інакше процедура A_1 закінчує роботу, при цьому із множини локальних екстремумів $\{\mu_{sj}^{*r}\}$ вибирається глобальний μ_{sj}^{**r}, що відповідає оптимальному розв'язку задачі (2).

Для зниження тимчасової складності роботи алгоритму можливо використовувати однопрохідний варіант реалізації даної процедури на основі стягнутого дерева шляхів, наведеного на рис. 4.8. При цьому однопрохідна процедура A_2 має такий вигляд, як наведено нижче.

Процедура A_2 з 1 проходом

Крок 1. З вершини s будуються всі можливі шляхи рангу $r = 1$ до всіх вершин графа D (рис. 4.8), що задовольняють обмеження $g_j(X_1, X_2,..., X_n)$; $j = \overline{(1,m)}$, при цьому довжини по вагах функціонала й обмежень обчислюються відповідно до співвідношень (4.6).

Крок 2. Використовуючи шляхи поточного рангу r, будуються всі можливі шляхи рангу $r = r + 1$, що задовольняють обмеження

$g_j(X_1, X_2, ..., X_n)$; $j = \overline{(1, m)}$ з використанням рекурентного співвідношення (4.7). При цьому перевірка обмежень і вибір шляху, максимального по вагах функціонала, здійснюється на основі обчислень довжин шляхів по вагах функціонала й обмежень відповідно до співвідношень (4.6). (Слід зазначити, що якщо в процесі застосування рекурентного співвідношення (4.7) виникають кілька шляхів однакової довжини, то необхідно їх усе продовжувати на наступному ранзі.)

Крок 3. Перевіряємо $m_{sj}^{r+1} = \varnothing$, якщо так, то шлях μ_{sj}^{*r} максимальної довжини, отриманий на ранзі *r*, є локальним екстремумом розв'язуваної задачі, інакше виконуємо наступний крок.

Крок 4. Перевіряємо ранг $r = n$, якщо ні, то переходимо до виконання кроку 2, інакше процедура A_2 закінчує роботу, і шлях μ_{sj}^{*r} максимальної довжини, отриманий на ранзі $r = n$, відповідає оптимальному розв'язку задачі.

Ще одним варіантом зменшення тимчасової складності алгоритмів на основі процедур A_1 і A_2 можуть бути процедури $A^{/}$ і $A^{//}$, що відрізняються від A_1 і A_2 тим, що на кожному ярусі формування шляхів на основі процедур A_1 і A_2 локальні екстремуми будуть виділятися не в кожній множині, а виділятимуться глобальні екстремуми на ярусі, і на основі шляху, що відповідає глобального екстремуму на ярусі, формуються шляхи наступного ярусу, що задовольняють обмеження, при цьому процедури $A^{/}$ і $A^{//}$ будуть мати вигляд, як наведено нижче.

Процедура $A^{/}$

Перед початком роботи процедури $A^{/}$ змінна i: = 1.

Крок 1. Змінна $s = i$ і з вершини s будуються всі можливі шляхи рангу $r = 1$ до всіх вершин графа *D* (рис. 4.9), що задовольняють обмеження $g_j(X_1, X_2, ..., X_n)$; $j = \overline{(1, m)}$, при цьому довжини по вагах функціонала й обмежень обчислюються відповідно до співвідношень (4.6). Далі виділяється найдовший шлях на ярусі.

Крок 2. Використовуючи найдовший шлях поточного рангу *r*, побудований на попередньому кроці, будуються всі можливі шляхи рангу $r = r + 1$, що задовольняють обмеження $g_j(X_1, X_2, ..., X_n)$; $j = \overline{(1, m)}$ з використанням рекурентного співвідношення (4.7). При цьому перевірка обмежень і вибір шляху, максимального по вагах функціонала, здійснюється на основі обчислень довжин шляхів по вагах функціонала й обмежень відповідно до співвідношень (4.6).

Крок 3. Перевіряємо $m_{sj}^{r+1} = \varnothing$, якщо так, то шлях μ_{sj}^{*r} максимальної довжини, отриманий на ранзі r, є локальним екстремумом розв'язуваної задачі, інакше виконуємо наступний крок.

Крок 4. Перевіряємо $i = n - 1$, якщо ні, то $i = i + 1$ і переходимо до виконання кроку 1, інакше процедура $A^{/}$ закінчує роботу, при цьому із

множини локальних екстремумів (отриманих за один прохід) $\{\mu_{sj}^{*r}\}$ вибирається глобальний μ_{sj}^{**r}, що відповідає оптимальному розв'язку задачі.

Процедура A''

Крок 1. З вершини s будуються всі можливі шляхи рангу $r = 1$ до всіх вершин графа D (рис. 4.8), що задовольняють обмеження $g_j(X_1, X_2, ..., X_n)$; $j = \overline{(1, m)}$, при цьому довжини по вагах функціонала й обмежень обчислюються відповідно до співвідношень (4.6). Далі виділяється найдовший шлях на ярусі.

Крок 2. Використовуючи найдовший шлях поточного рангу r, побудований на попередньому кроці, будуються всі можливі шляхи рангу $r = r + 1$, що задовольняють обмеження $g_j(X_1, X_2, ..., X_n)$; $j = \overline{(1, m)}$ з використанням рекурентного співвідношення (4.7). При цьому перевірка обмежень і вибір шляху, максимального по вагах функціонала, здійснюється на основі обчислень довжин шляхів по вагах функціонала й обмежень відповідно до співвідношень (4.6).

Крок 3. Перевіряємо $m_{sj}^{r+1} = \varnothing$, якщо так, то шлях μ_{sj}^{*r} максимальної довжини, отриманий на ранзі r, є локальним екстремумом розв'язуваної задачі, інакше виконуємо наступний крок.

Крок 4. Перевіряємо ранг $r = n$, якщо ні, то переходимо до виконання кроку 2, інакше процедура A'' закінчує роботу, і шлях μ_{sj}^{*r} максимальної довжини, отриманий на ранзі $r = n$, відповідає оптимальному розв'язку задачі.

Оцінка складності процедур {A}

Оскільки кількість шляхів, що будуються на довільному ранзі r, не може перевищити $(n-1)*(n-1)$, максимальний ранг r довільного шляху не перевищує $(n-1)$, а кількість циклів, виконуваних процедурою A_1, дорівнює n, то після n циклів кількість шляхів, що побудує процедура A_1, не може перевищити $(n-1)*(n-1)*(n-1)*n \approx n^4$, а кількість оброблених векторів n^5. З урахуванням кількості, що складаються (k) у функціоналі, і кількості обмежень (m) тимчасова складність алгоритму не перевищить у найгіршому разі $O(n^5 (m + 1))$. У випадку, коли розв'язання задачі здійснюється за один прохід процедури A_2 або за один прохід процедури A'', але з виділенням найдовшого шляху, на ярусі складність процедур A_2 і A'' не перевищить відповідно $O(n^4 k(m + 1))$ і $O(n^3 k(m + 1))$. Отже, алгоритми A_5, A_4, A_3 мають відповідно тимчасову складність, що не перевищує в найгіршому разі $O(n^5 k(m + 1))$, $O(n^4 k(m + 1))$ і $O(n^3 k(m + 1))$. Таким чином, запропонований підхід розв'язання довільних задач булевого програмування дозволяє на основі запропонованих алгоритмів розв'язувати з єдиних позицій будь-які задачі лінійного й нелінійного програмування за поліноміальний час із необхідною точністю. Якщо в співвідношення (4.1) увести позначення

$$f_k(x) = \sum_{j=1}^{p_k} C_{kj} S_k(C_n^k),$$

то воно набуде вигляду

$$F(x) = \sum_k f_k(x).$$

У загальному випадку, якщо функціонал і обмеження являють собою довільну нелінійну функцію від $f_k(x)$, то й така задача нелінійного програмування також може бути ефективно розв'язана за допомогою запропонованого підходу, і при цьому будуть змінюватися тільки правила ваг вершин клік.

4.3. Метод розв'язання задачі визначення найкоротших гамільтонових шляхів

Ми розглядали задачі булевого програмування, у яких коефіцієнти C_{lj} в (1) не залежать від порядку проходження змінних, а тільки від їхніх поєднань. Покажемо на прикладі задачі про найкоротший гамільтонів шлях, що на основі запропонованої процедури $\{A\}$ може ефективно розв'язуватися, і задачі булевого програмування, у якій довжина шляху в графі G залежить не тільки від поєднання змінних, але й від перестановок змінних.

Процедура А для визначення найкоротшого гамільтонового шляху в довільному графі

З вершини *s* будуються всі можливі шляхи рангу *r* = 1 до всіх вершин графа D, далі, використовуючи шляхи рангу *r* = 1, будуються всі можливі шляхи рангу *r* = 2 і на їхній основі формуються шляхи наступного рангу з використанням рекурентного співвідношення (3.35) і т. д. до побудови шляхів рангу r = n – 1.

$$\mu_{sp}^{r+1} = \min_j \{\{\mu_{sj}^r\} \cup (j,p)\}; j = \overline{(1,n)}; p = \overline{(1,n)}; j \neq p. \qquad (4.8)$$

(Слід зазначити, що якщо в процесі застосування рекурентного співвідношення (4.8) виникають кілька найкоротших шляхів однакової довжини, то необхідно їх усі будувати на наступному ранзі.)

Можна виділити такі особливості роботи процедури *A*. У процесі її роботи може виникати дві ситуації. Перша, коли процедура *A* на кожному кроці побудувала шляхи в множині m_{sj}^r, тобто принцип оптимальності роботи процедури не порушувався, і друга, коли до однієї з вершин

жодного шляху побудувати не можна. Остання обставина можлива у двох випадках:

а) якщо аналізований граф неповний і до вершини p не існує шляху деякого рангу $r = k$;

б) коли деяка вершина p увійде в усі шляхи попереднього рангу.

Неважко побачити, що в першій ситуації процедура А не втрачає оптимального розв'язку, а в другій ситуації (випадок б) оптимальний розв'язок може бути загублений, оскільки принцип оптимальності роботи процедури порушується. Це означає, що на основі процедури А ми можемо побудувати тільки наближений алгоритм розв'язання задачі. Однак після порушення принципу оптимальності наступне продовження шляхів з використанням рекурентного співвідношення (4.8) дозволить мінімально відхилитися від оптимального розв'язку.

Алгоритм A_1 розв'язання задачі визначення найкоротших гамільтонових шляхів у довільних графах

Крок 1. Привласнюємо значення $S = 1$ змінній S і в графі D, використовуючи процедуру A, визначаємо найкоротші шляхи рангу $r = n$ -1 від вершини S до всіх інших вершин графа й збільшуємо значення $S = S$ +1.

Крок 2. Перевіряємо $S = n$, якщо ні, то переходимо до виконання кроку 1, інакше виконуємо наступний крок.

Крок 3. Серед усіх побудованих найкоротших шляхів рангу $r = n$ - 1 на кроках 1 і 2 вибираємо найкоротший, і алгоритм закінчує роботу.

Для розв'язання задачі за один прохід визначимо точку входу, починаючи з якої почне роботу алгоритм. Для цього введемо поняття мінімально вилученої вершини і в довільному графі $G(V, E)$.

Нехай задано граф G(V, E) з n вершинами й множиною ребер E, кожному з яких привласнено вагу у вигляді довільного додатного числа β_i . Поставимо кожній вершині s графа $G(V, E)$ у відповідність вектор V_s

$$X_s = \{\beta_1, \beta_2, \dots, \beta_s, \wp, \beta_{s+1}, \dots, \beta_n\}; \quad s = \overline{(1,n)},$$

у якому на s-й позиції стоїть символ $\wp$, що означає, що елемент β_s у X_s відсутній. Множина $\{\beta_i\}$ являє собою множину додатних чисел, до яких будемо відносити й 0. *Будемо говорити, що деяка вершина і є мінімально віддаленою від вершини s, обумовленої вектором X_s, якщо відстань $d(s,i)$ між вершинами s і I визначається співвідношенням*

$$d(s,i) = \min_i \{\beta_i \in X_s\}. \tag{4.9}$$

Якщо існує Q вершин і, що задовольняють співвідношення (4.9), то це означає, що існує Q, що мінімально дорівнює віддаленим вершинам від вершини s, обумовленої вектором X_s. Визначивши вершину і входу для

графа, можна розв’язувати задачу визначення найкоротшого гамільтонового шляху за один прохід процедури A_1, при цьому похибка порівнянно з n-прохідною процедурою, як буде показано нижче, збільшується не більш ніж на один відсоток, а складність алгоритму знижується в n – 1 раз. Алгоритм із використанням мінімально вилученої точки позначимо A_2.

4.4. Експериментальне дослідження алгоритмів розв’язання задач булевого програмування на основі теорії графів

При експериментальному дослідженні розроблених алгоритмів основна увага приділена оцінці тимчасової складності алгоритмів, оцінці похибки наближених алгоритмів і порівняльному аналізу розроблених алгоритмів з відомими алгоритмами.

Експериментальне дослідження алгоритмів на основі розроблених процедур для загальних методів розв’язання задач булевого програмування

Досліджувалися такі алгоритми: алгоритм A_5 на основі багатопрохідної процедури A_1, алгоритм A_4 на основі однопрохідної процедури A_2 і алгоритм A_3 на основі однопрохідної процедури A''. При дослідженні коефіцієнти у функціоналі й обмеженнях генерувалися за рівномірним законом розподілу у функціоналі в діапазоні від 0 до 10, а в обмеженнях від 0 до 20. На кожну точку при оцінці тимчасової складності алгоритмів у середньому й похибки алгоритмів розв’язувалося не менше 50 тестових задач, результати отримані з довірчою ймовірністю 0,95. Як точний алгоритм використовувався розроблений алгоритм для задачі квадратичного й лінійного програмування, скомбінований на основі використання для прогнозування ідей рангового підходу, а для відсікання безперспективних шляхів методу галузей і границь, що дозволив зняти похибки для задач до розмірності, що не перевищує $n = 70$. Графіки залежності похибки від розмірності (n) розв'язуваних задач і від кількості обмежень (m) у задачі (4.2) наведені на рис. 4.10 - 4.12, з яких видно, що похибка алгоритмів зі збільшенням кількості обмежень m асимптотично зменшується, зі збільшенням n зростає й $m \geq 50$, похибка алгоритмів стабілізується й для задач лінійного програмування не перевищує 2 %, а для задач квадратичного 5-10 %.

Розв’язання тестових задач показало, що збільшення діапазону зміни коефіцієнтів у функціоналі й обмеженнях призводить до різкого зниження похибок алгоритмів, перехід від нелінійностей одного порядку до нелінійностей більш високого порядку може призводити до незначного зростання похибок при невеликій кількості обмежень, а зі зростанням кількості обмежень зростання похибки дуже швидко компенсується. Експериментальне дослідження тимчасової складності показало (рис. 4.13), що кількість оброблюваних векторів не залежить від кількості

обмежень і в середньому для алгоритмів A_5, A_4, A_3 тимчасова складність не перевищує відповідно $O(0{,}1n^{4,9})$, $O(0{,}3n^{3,7})$ і $O(0{,}4n^{2,8})$.

а

б

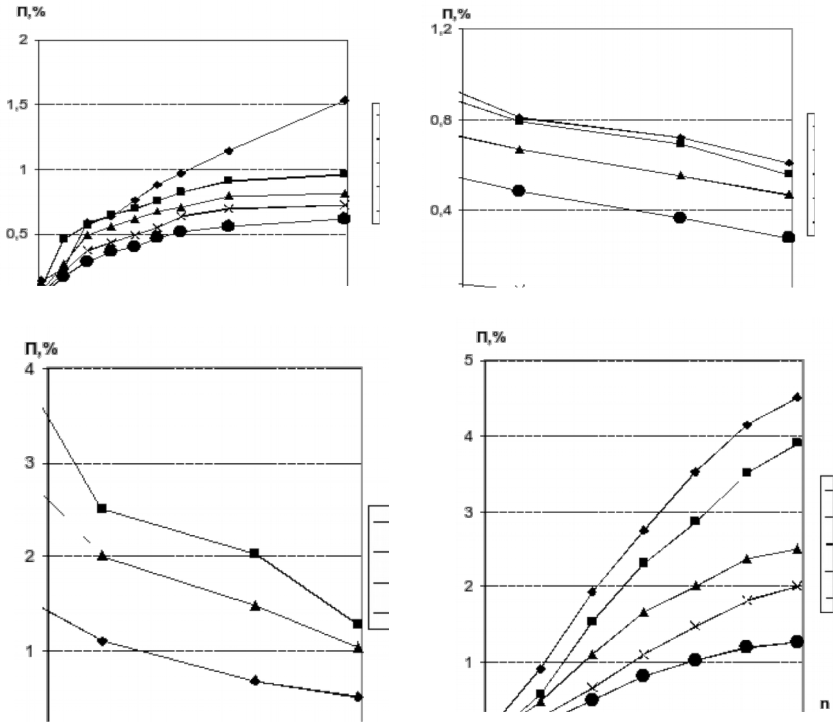


Рис. 4.10. Залежність похибки алгоритму A_5 від розмірності розв'язуваної задачі: лінійного булевого програмування (n) при різній кількості обмежень (m) (а); квадратичного булевого програмування (n) при різній кількості обмежень (m) (б)

а

б

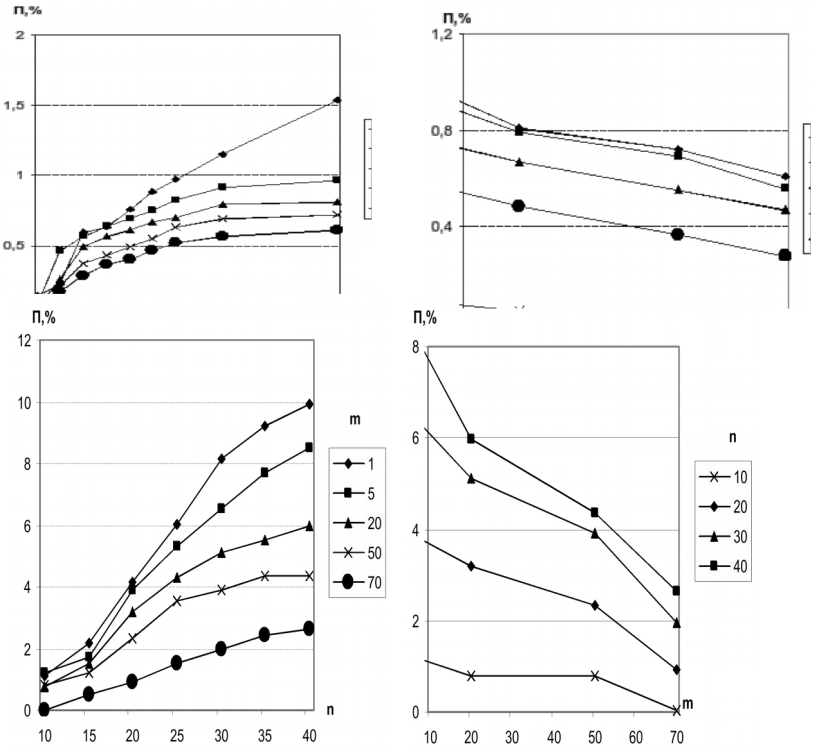


Рис. 4.11. Залежність похибки алгоритму А4 від розмірності розв'язуваної задачі: лінійного булевого програмування (n) при різній кількості обмежень (m) (а); квадратичного булевого програмування (n) при різній кількості обмежень (m) (б)

а

б

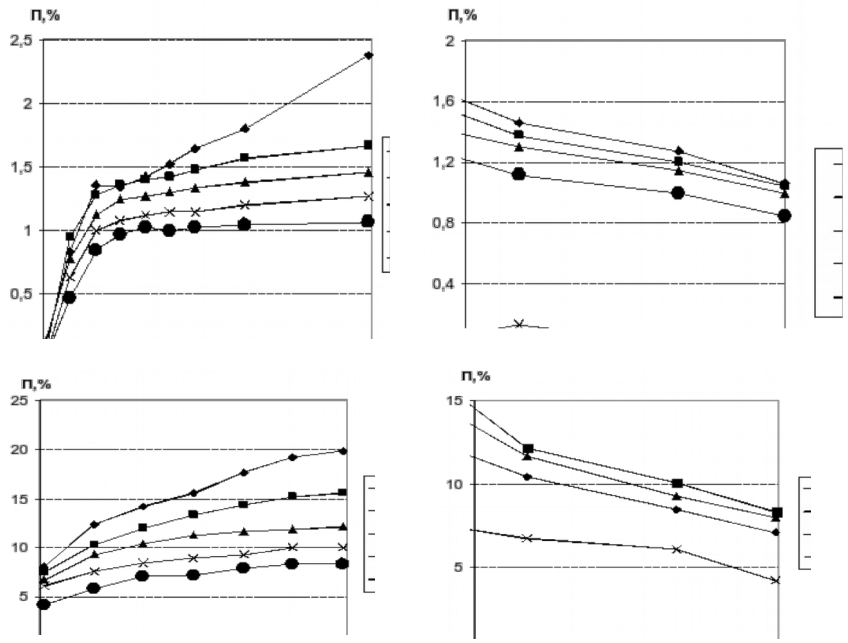


Рис. 4.12. Залежність похибки алгоритму A_3 від розмірності розв'язуваної задачі: лінійного булевого програмування (n) при різній кількості обмежень (m) (а); квадратичного булевого програмування (n) при різній кількості обмежень (m) (б)

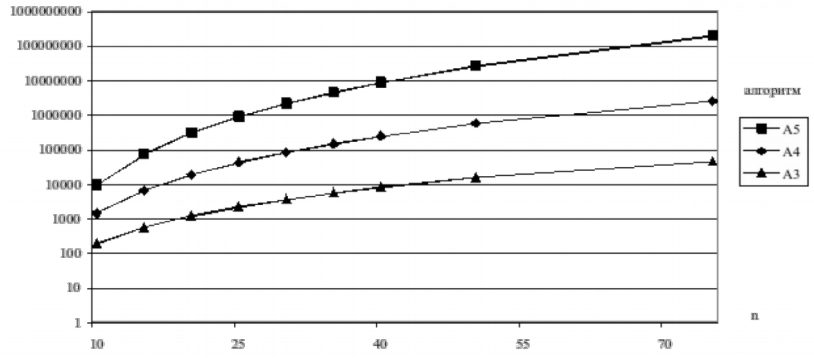
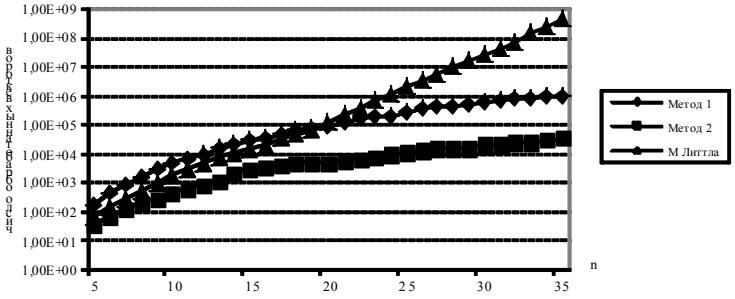


Рис. 4.13. Залежність кількості оброблюваних векторів від розмірності розв'язуваної задачі лінійного й квадратичного програмування для алгоритмів A_5, A_4, A_3

Експериментальне дослідження алгоритмів розв'язання задачі визначення найкоротших гамільтонових шляхів

При експериментальному порівнянні розроблених алгоритмів з відомими ваги ребер графа генерувалися за рівномірним законом розподілу в діапазоні (0–50). Для одержання середнього значення кожної точки графіків всіх аналізованих характеристик розв'язувалося по 200 тестових задач, всі результати статистичного аналізу отримані з довірчою ймовірністю 0,95. На всіх рисунках розроблено алгоритми A_1 і A_2, позначені як метод –1 і метод –2. Як видно з графіків, наведених на рис. 4.14, при $n \geq 27$ алгоритми Літтла й локального пошуку мають істотно більш високу тимчасову складність порівняно з розробленими. При цьому, як показано в роботах [85–98], алгоритми локального пошуку мають похибку, що лежить у діапазоні від 7 до 29 % і зростає зі збільшенням розмірності розв'язуваних задач. У розроблених алгоритмах на основі ідей рангового підходу похибка зі збільшенням розмірності задачі стабілізується й не перевищує 2 % (рис. 4.15). Із графіків, наведених на рис. 4.16, видно що відсоток неточних розв'язків для алгоритму A_1 при $n \geq 27$ дуже низький. Експериментальне дослідження показало, що при $n \geq 27$ на 2000 тестових задач у середньому тільки 3 давали наближений розв'язок і при цьому похибка не перевищувала 1–2 %. Використання евристичного правила в алгоритмі A_2 дозволило, з одного боку, істотно зменшити тимчасову складність цього алгоритму порівняно з A_1, але, з іншого боку, при цьому зростає похибка, і кількість неточних розв'язків у ньому зі збільшенням розмірності розв'язуваної задачі зростає (рис. 4.16).



а б

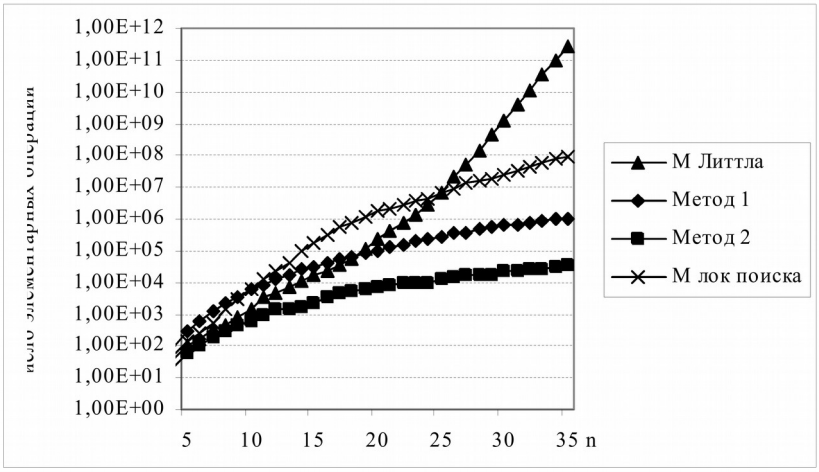


Рис. 4.14. Залежність кількості оброблюваних векторів від розмірності розв'язуваної задачі (а) і елементарних операцій (б)

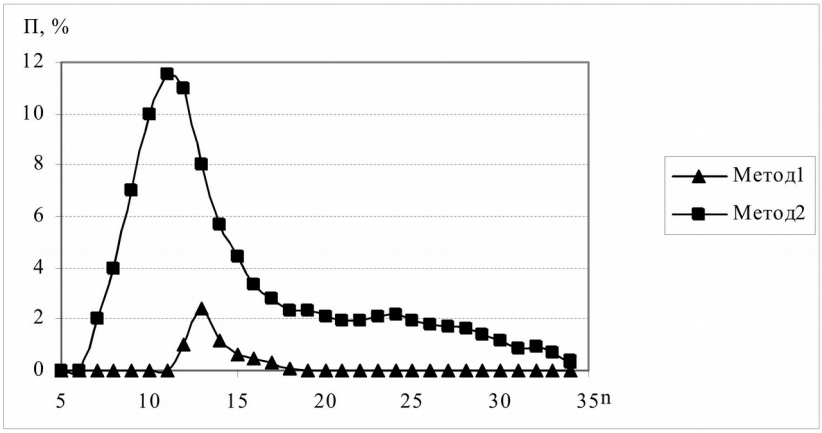


Рис. 4.15. Залежність відносної похибки від розмірності розв'язуваної задачі

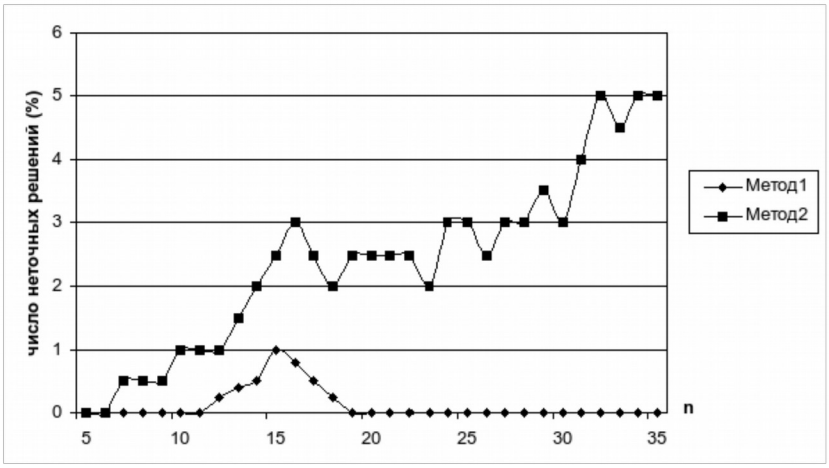


Рис. 4.16. Залежність кількості неточних розв’язків від розмірності розв'язуваної задачі

4.5. Управління реалізацією робочим навантаженням у СБД

Як загальний підхід розв’язання задачі управління реалізацією робочого навантаження в СБД вироблено два основних напрямки. Перший напрямок пов'язаний з оптимізацією структури даних у мережних базах даних. Другий напрямок пов'язано з оптимізацією процесу управління множиною транзакцій і запитів при їхній реалізації в мережних базах даних.

У період з 1990 по 2002 рік по першому напрямку виконано багато робіт і практичних розробок, у яких почато спробу піти від сформованих протиріч. Але, як правило, кожний знайдений оптимальний розв’язок організації структури даних у зв'язку з ускладненням методів доступу спричиняв породження нових проблем в організації процесу управління множиною запитів і транзакцій.

Розв’язанню задач організації процесу управління множиною транзакцій і запитів при їхній реалізації в мережних базах даних приділялося значно менше уваги. Це пов'язане з тим, що:

- не існує єдиної методології проектування й реорганізації множини запитів і транзакцій;
- існуючі наближені алгоритми, що реалізують методи управління, не мають прийнятної точності знайденого розв’язку;
- дані задачі за своєю природою є NP-повними і ефективними точними алгоритмами, розв’язку для них не існує;
- реалізація даних алгоритмів вимагає значних системних ресурсів, а їхня вартість за умови набору необхідного об'єму обчислювальної пам'яті значно перевищує вартість витрат на розроблення методів розв’язання першої задачі.

Незважаючи на це, актуальність одержання розв’язку відповідно до другого підходу залишається досить високою, що в першу чергу пов'язано зі зростанням обсягів інформації й збільшенням кількості користувачів мережних баз даних.

Як випливає з проведеного аналізу (див. п. 1.2), управління реалізацією робочого навантаження складається з ряду взаємозалежних процесів, реалізованих у СБД. Аналіз цих процесів показав, що найбільш істотний вплив на якість і оперативність управління має процес формування графіка реалізації множини транзакцій і запитів користувачів. У свою чергу вивчення даного процесу дозволило виділити дві основні задачі, що потребують свого розв’язання й формування і є складовою процесу, графіка реалізації множини транзакцій і запитів користувачів:

- задача підготовки вихідних даних;
- задача складання плану реалізації запитів користувачів і транзакцій.

Нехай є множина запитів і транзакцій $Q=\{q_j\}$, $i=\overline{1,m}$ до мережної бази даних, що містить множину $\{a_j\}$, $j=\overline{1,n}$ атрибутів, що відображають предметну область цієї інформаційної структури. Кожний запит q_i для пошуку релевантних об'єктів вимагає обробки частини даних інформаційної структури, відображуваної підмножиною $\{a_{ik}\}\subseteq\{a_j\}$, $k=\overline{1,r_i}$, причому кожний атрибут a_{ik} перевіряється на множині умов $\{L_{ikl}\}$, $l=\overline{1,r_i}$, які можуть входити в різні логічні вирази семантичних фрагментів запиту. При цьому семантика запиту й логічна структура організації даних висувають множину вимог $R=\{r_{ikl}\}$ до організації процесу обробки даних. Кожна вимога R_{ikl} являє собою обмеження на порядок перевірки умов, що полягає в обов'язковості попередньої перевірки певної підмножини умов, що входять у множини $\bigcup_k\bigcup_l L_{ikl}$, перед перевіркою умови L_{ikl}. Структури запиту й транзакцій вступники на обслуговування подамо у вигляді графа

$G(V,E)$, де множині вершин відповідають елементарні оператори транзакцій і запитів. Уведені вершини з'єднані ребрами, якщо, за логічних умов, оператори впливають на виконання наступних операторів транзакцій і запитів. Ребрам графа зіставимо вагові характеристики, що являють собою суму часу доступу до необхідного атрибута $\mathbf{a_j}$ і час перевірки логічних умов L_{ikl} для реалізації оператора запиту (транзакції) q_i у базі даних:

$$T_{\text{real.}} = \sum_{i}^{n} \sum_{k}^{m} (T_{ik}^{D} + T_{ik}^{o}) . \qquad (4.10)$$

Оскільки час доступу до необхідного атрибута $T_{ik}^{Ä}$ залежить від характеристик використовуваної СУБД, організації структури зберігання мережної бази даних і використовуваних методів доступу, то його можна визначити до початку виконання процесу реалізації транзакцій і запитів. Друга ж частина вагової характеристики T_{ik}^{o} залежить від логічних умов пропонованих користувачами в семантичній частині запитів і транзакцій, а оскільки заздалегідь немає можливості визначити порядок появи тих або інших логічних умов перевірки атрибута a_j, то зазначена тимчасова характеристика є змінюваною і такою, що потребує динамічної оцінки в процесі розв'язання задачі безпосередньо. Тому виникає потреба в розв'язанні задачі підготовки вихідних даних для визначення вагових характеристик ребер графа $G(V,E)$. Фізично ця задача реалізується за допомогою створення в оперативній пам'яті ЕОМ матриць, що відображують зважений граф набору транзакцій і запитів, попередній аналіз яких проведено за допомогою застосування теорії реляційної алгебри.

Другим етапом побудови графіка реалізації множини транзакцій і запитів є безпосередньо формування плану виконання операторів ЯМД із урахуванням вимог, пропонованих користувачами.

Оскільки час реалізації вступників на обслуговування запитів і транзакцій істотно залежить від порядку їхнього виконання, виникає необхідність зазначення порядку їхньої обробки, що мінімізує сумарний час реалізації робочого навантаження.

Вихідними даними для розв'язання зазначеної задачі виступає множина пар $\left\{(a_{ik}^{d}, L_{ik}^{d})\right\}$, отриманих у результаті розв'язання першої задачі, і вершинами, що є, графа плану реалізації операторів запитів і транзакцій, а також вагові характеристики ребер, що з'єднують вершини даного графа й фізично відображають час перевірки логічної умови накладається користувачем для реалізації семантичного фрагмента q_i запиту. Тоді розв'язання задачі досягається мінімізацією цільової функції, що являє собою час обслуговування множини запитів Q, який має вигляд

$$F = \sum_{i=1}^{m} \sum_{k=1}^{r_i} V_{ik} \left[\left[\frac{1}{W_{ik}} \right] T_{ik}^{A} P_{ik}^{(1)} + T_{ik}^{0} \sum_{i=1}^{p_{ik}} P_{ikl}^{(2)} \right] \rightarrow \min , \qquad (4.11)$$

де V_{ik} – кількість екземплярів атрибута a_{ik};

$P_{ik}^{(1)}$ і $P_{ilk}^{(2)}$ – коефіцієнти використання a_{ik}, що характеризують імовірності використання екземплярів атрибута a_{ik} в процесах доступу й обробки відповідно;

T_{ik}^{A} – середній час доступу до одного екземпляра атрибута a_{ik};

T_{ik}^{o} – середній час обробки a_{ik} (перевірки накладеного на a_{ik} умови або обробки - a_{ik} як аргумент функції);

W_{ik} – загальна (по всіх запитах) кількість умов, що перевіряються одночасно на a_{ik}.

Коефіцієнти $P_{ik}^{(1)}$ й $P_{ik}^{(2)}$ визначаються за формулами

$$P_{ik}^{(1)} = \sum_{z=1}^{V_{ik}} X_{ikz} / V_{ik} , \qquad (4.12)$$

$$P_{ilk}^{(2)} = \sum_{z=1}^{V_{ik}} Y_{iklz} / V_{ik} , \qquad (4.13)$$

де X_{ikz} – булева змінна, що набуває значення 0 тільки тоді, коли обробка a_{ikz} по всіх p_{ik} умовах при поточних значеннях даних загубила свій зміст (наприклад, якщо запис при обслуговуванні попереднього запиту було вилучено);

Y_{iklz} – булева змінна, що набуває значення 0 тільки тоді, коли обробка a_{ikz} за *l-ю* умовою при поточних значеннях даних загубила свій зміст.

У виразі (4.11) V_{ik} визначається структурою інформаційної системи, її тимчасовими характеристиками й використовуваним програмним апаратом обробки інформації й може бути визначена до обробки запиту. Оптимізація досягається складанням такої впорядкованої множини пар $\left\{ \left(a_{ik}^{d}, L_{ik}^{d} \right) \right\}$, при якому вираз (4.11) досягав би мінімального значення. Побудова такої впорядкованої множини визначається двома факторами: послідовністю обходу a_{ik} в інформаційній структурі і вибором множини умов, що перевіряються одночасно для кожного a_{ik}.

Назвемо *и*-м планом реалізації множини запитів в інформаційній структурі впорядковану множину пар $S_u = \left\{ \left(a_{ik}^{d}, L_{ik}^{d} \right) \right\}$, де $d = \overline{1, N_u}$, N_u — номер кроку виконання плану S_u (на якому обробляється єдиний атрибут);

$a_{ik}^d \in \{a_j\}$ – атрибут, що на d-му кроці піддається перевірці одночасно на множині L_{ikl}^d умов так, що $L_{ik}^d \subseteq \{L_{ikl}\}$ й $L_{ik}^1 \cap L_{ik}^2 \cap ... \cap L_{ik}^{N_u} = \varnothing$.

Опишемо множину S припустимих планів перегляду $S_u \in S$, якщо $\forall d$ й $\forall L_{ikl} \in L_{ik}^d$ є справедливим: $\{R_{ikl}\} \subseteq L_{ik}^1 \cup L_{ik}^2 \cup ... \cup L_{ik}^d$, тобто для кожного перевіряється у відповідності з S_u умова виконання необхідної вимоги з *R*. Цільова функція (4.11) є тимчасовою оцінкою виконання плану S_u на етапі диспетчеризації. Таким чином, задача зводиться до знаходження оптимального плану S^* такого, що

$$T(S^*) = \min T(S_u), \tag{4.14}$$

де $T(S_u) = F$.

Таким чином, друга задача може розглядатися як задача визначення найкоротшого гамільтонового шляху в графі G. Оскільки дана задача належить до класу NP-повних задач, а її розв'язання повинне здійснюватися в реальному масштабі часу, виникає необхідність у розробленні методу, що дозволяє знаходити розв'язок при заданих обмеженнях на тимчасові ресурси. Крім того, не менш важливою вимогою, висунутою до алгоритмів реалізованого методу, пошуку найкоротшого гамільтонового шляху, є точність, оскільки кількість збоїв і колізій у системі залежить від точності сформованого графіка й здатності виявлення тупикових ситуацій реалізації множини транзакцій і запитів. Загальні методи розв'язання даної задачі для динамічних інформаційних систем є важкозастосовними через високі вимоги до точності знаходження оптимального розв'язку й швидкості її розв'язання через значну розмірність. Тому для її розв'язання доцільно використовувати метод розв'язання з у п. 4.3 на основі вироблених загальних підходів до розв'язання задач булевого програмування. Як видно з графіків, наведених на рис. 14 – 16, алгоритми, засновані на ідеях рангового підходу, дозволяють на порядок швидше й точніше розв'язувати задачу управління реалізацією робочого навантаження в СБД порівняно з методами, заснованими на ідеях методу галузей і границь.

Вправи

1. Поясніть, чому запропоновані в даному розділі узагальнені процедури можна розглядати як універсальну схему розв'язання довільних задач комбінаторної оптимізації.

2. Поясніть, як обчислюються довжини шляхів у графі при розв'язанні довільних задач нелінійного програмування.

3. Поясніть, чому зі збільшенням кількості обмежень при розв'язанні задач нелінійного програмування похибка розв'язків зменшується.

4. Розв’яжіть приклад 3.1 розд. 3 алгоритмом для розв’язання задач нелінійного програмування.

5. Поясніть, як задача управління навантаженням СБД може бути зведена до задачі найкоротших гамільтонів шляхів.

6. Покажіть, що алгоритм визначення найкоротшого гамільтонового циклу, розглянутий у даному розділі, наближений.

7. Чому в узагальненій процедурі розв’язання довільних задач використовується стягнуте дерево всіх шляхів?

РОЗДІЛ 5. Організація паралельних обчислень при розв’язанні задач дискретної оптимізації

5.1. Ранговий підхід до організації паралельних обчислень

5.1.1. Проблеми побудови паралельних алгоритмів і обчислювальних систем для задач дискретної оптимізації

В автоматизованих системах управління для виконання найбільш складних розрахунків при розв’язанні задач управління створюються суперпроцесори різних орієнтацій: матричні, мікроконвеєрні, поточно-векторні й ін., до яких висуваються вимоги, що значною мірою відрізняються від вимог до звичайної обчислювальної техніки. Для комп'ютеризації цієї сфери необхідним є створення ефективних і надійних засобів контролю й управління процесами, що швидко ~~процесами~~ протікають, збільшення інтелекту в пристроях зв'язку з об'єктом і оператором, будь-яке зменшення габаритів, споживаної потужності й т.п. У перспективі ставиться завдання створення технічних засобів, що заміщають якомога повніше людину на базі кібернетичних систем. Необхідність здійснювати управління процесами, що швидко протікають, при управлінні складними космічними засобами й засобами наземного базування призвело до розширення діапазонів сигналів, що надходять від датчиків, і зменшення часу, що відпускається на їхню обробку. Управління такими об'єктами й процесами висуває до технічних засобів настільки високі вимоги щодо оперативності, продуктивності й ефективності, що вони не забезпечуються в рамках традиційних методів перетворення й обробки інформації й принципів побудови універсальних ЕОМ [188]. Створення ефективних засобів високої оперативності й продуктивності, що

забезпечують перетворення й обробку інформації в реальному масштабі часу, вимагає розроблення теоретичних основ їхньої побудови, у тому числі розгляду процесів перетворення обробки інформації з погляду поєднання їх у часі, питань спеціалізації або проблемної орієнтації технічних засобів, розпаралелювання обчислювальних процесів, використання методів апаратної реалізації алгоритмів. Аналіз основних тенденцій розвитку паралельних обчислювальних систем показує, що тут можна виділити такі напрямки розвитку ~~паралельних обчислювальних систем (~~ПОВС~~)~~ [224]:

по-перше, створення ~~ПВС~~ ПОС для розв'язання задач первинної обробки інформації на нижньому рівні систем, які зводяться до масштабування, перемножування, диференціювання, інтегрування, фільтрації, перетворення сигналів;

по-друге, розроблення проблемно~~-~~орієнтованих ~~ПВС~~ ПОС для розв'язання задач лінійної алгебри, а також диференціальних рівнянь у похідних;

по-третє, розроблення ~~ПВС~~ ПОС для розв'язання задач комбінаторної оптимізації й теорії графів. Цей напрямок є найбільш складним і найменш розробленим на сьогоднішній день, а також найбільш важливим з погляду побудови систем динамічного управління в мережах АСУ й автоматизації процесу ухвалення рішення.

Особливістю організації паралельних обчислень, як показано в роботі [224], є взаємозв'язок архітектури паралельної обчислювальної системи й структури алгоритму, реалізованого на даній архітектурі.

5.1.2. Організація паралельних обчислень на основі рангового підходу

Можна виділити два типи задач, які доводиться розв'язувати при розробленні ~~ПВС.~~ ПОС.

Задача 1. Дан~~ий~~ орієнтований граф, що описує обчислювальний алгоритм і зазначен~~ий~~ специфікація операцій, що ототожнюються з його вершинами. Відома номенклатура пристроїв, що ~~в~~у сукупності виконують всі операції алгоритму. Необхідно в межах виділених лімітів вибрати склад функціональних пристроїв, побудувати орієнтований граф обчислювальної системи й указати програму або множину програм, що забезпечують реалізацію даного алгоритму за мінімальний час.

Задача 2. Дан~~ий~~ орієнтований граф, що описує обчислювальний алгоритм, і зазначена специфікація операцій, що ототожнюються з його вершинами. Дан~~ий~~ орієнтований граф, що описує обчислювальну систему, і зазначена специфікація пристроїв, що ототожнюються з його вершинами. Необхідно відповістити на запитання, чи можна реалізувати алгоритм на даній системі, у випадку позитивної відповіді вказати програму або множину програм, що забезпечують реалізацію алгоритму за мінімальний час.

Обидві задачі називають задачами відображення алгоритмів на обчислювальні системи й належать до класу NP-пНовних.

Як показано в роботі [93], спроби розпаралелювання алгоритмів розв'язання оптимізаційних задач на основі ідей методу галузей і ~~границь~~ меж стикаються ~~і~~з проблемою визначення оптимальної кількості процесорних елементів, на якому доцільно розв'язувати оптимізаційну задачу. Збільшення кількості процесорних елементів n у ~~ПВС~~ ПОС алгоритми, що реалізує, на основі ідей методу галузей і ~~границь~~ меж призводить до зниження продуктивності системи, що обумовлено швидким збільшенням кількості обмінних операцій. Слід зазначити, що сама задача визначення оптимальної кількості процесорних елементів належить до класу NP-пНовних. У цьому плані вигідно відрізняються алгоритми, засновані на ідеях рангового підходу. Як буде показано далі, використання n процесорних елементів, що дорівнює кількості змінних в оптимізаційних задачах, дозволяє підвищити продуктивність системи в n раз. При цьому структура процесорних елементів, орієнтована на виконання операцій додавання, порівняння й об'єднання, виявляється досить простою і дозволяє будувати спеціалізовані обчислювачі на основі транспютерних технологій для розв'язання задач досить великої розмірності в масштабі реального часу.

5.2. Поняття циклічної паралельної обчислювальної системи й організація в ній обчислень

Математичний аналіз принципів побудови супер-ЕОМ і інших високопродуктивних систем значною мірою ускладнюється більшою їхньою розмаїтістю, численними поняттями й відсутністю формалізованих принципів дослідження. Незважаючи на велику кількість досліджень у цьому напрямку, картина, пов'язана з організацією паралельних обчислень, архітектурою паралельних обчислювальних систем, для різних типів задач не прояснилася. Найцікавішими в даному напрямку~~;~~ є роботи, виконані під керівництвом Г.І. Марчука [49], а також публікації В.В. Васильєва й Е.А. Ралдугина [52, 53, 127, 169]. Варто виділити монографії В.В. Воєводіна [40, 44], у яких зроблена спроба об'єднати дослідження обчислювальних методів і архітектур обчислювальних систем у єдиний процес і розглянути питання відображення проблем у роботі обчислювальної математики й архітектури ~~у~~ обчислювальних систем. У роботі [40] зроблено песимістичний висновок про те, що немає ніяких підстав сподіватися на одержання загальної задачі відображення проблем математики на архітектуру обчислювальних систем за допомогою деякого універсального методу. Висновок базується на тім, що алгоритми можна представляти ациклічними орієнтованими графами, а ряд задач, пов'язаних з аналізом можливості ефективної реалізації довільного алгоритму A_1, якому відповідає граф G_1, на обчислювальній системі, заданій графом G_2, належать до NP-повних задач. У даній роботі робиться спроба розглянути різні концепції

побудови паралельних обчислювальних систем як проблемно-орієнтованих, так і універсальних на базі циклічних паралельних обчислень.

Алгоритм можна подати у вигляді графів, при цьому природно покласти, що алгоритм і його правило, що описує, дозволяють визначити:

- множину змінних, у перетворенні яких полягає реалізація алгоритму;
- множину операцій, виконуваних у процесі реалізації алгоритму;
- відповідність, що показує, які результати виконання попередніх операцій є аргументами для кожної операції.

Нехай задано деякий алгоритм A. Його можна подати у вигляді підмножини алгоритмів $\{A_j\}$, (j = 1, n), які необхідно виконати в певній послідовності, заданій графом $G(V, E)$. В останньому кожна вершина відповідає алгоритму A_j, а кожне ребро між вершинами i-j, спрямоване від вершини i до j, існує тільки в тому випадку, якщо алгоритм A_j можна виконати тільки після реалізації алгоритму A_i. Наприклад, на рис. 5.1, а наведена структурна схема деякого алгоритму А, задана у вигляді графа $G(V, E)$, з якого видно, що для його реалізації спочатку потрібно виконати алгоритм A_1. Результати роботи цього алгоритму є вихідними даними для виконання алгоритмів A_2, A_3, A_4. Після їхнього виконання реалізується алгоритм A_5, ~~а,~~ алгоритм A_6 спрацьовує після алгоритмів A_3, A_4, а A_7 – після A_5 і A_6. Очевидно, що ~~число~~ кількість різних подань графів алгоритму А визначається множиною способів, використовуваних для розв'язання задачі. У принципі граф G (рис. 5.1, а) можна розглядати як структуру програмного виробу, що реалізує алгоритм A. Тоді $\{A_j\}$ – множина програмних модулів, з яких складається програма реалізації алгоритму A. Очевидно, що якщо в кожну вершину графа G (рис. 5.1, а) помістити процесор, орієнтований на виконання алгоритму A_j, то ми одержимо обчислювальну систему, що дозволяє реалізувати алгоритм A. Припустимо, що кожний процесор $П_j$ виконує алгоритм A_j відразу, як тільки на його вхід надійдуть всі необхідні дані.

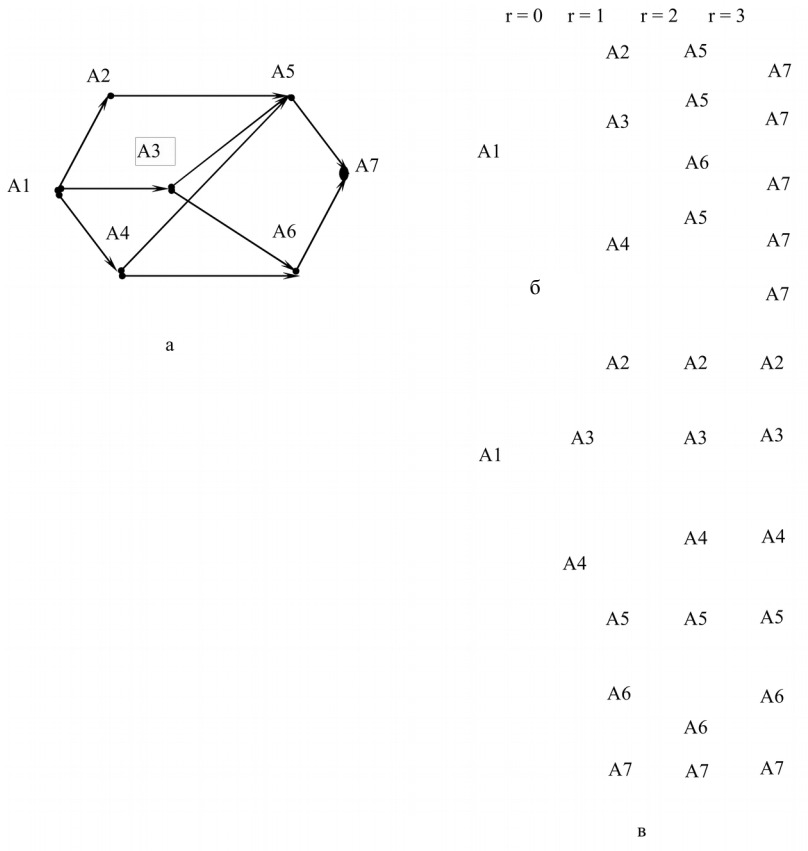


Рис. 5.1. ~~б –~~ Графи виконання алгоритмів

Позначимо t_j – час реалізації кожного з алгоритмів A_j на відповідному процесорі $П_j$. Тоді, якщо привласнити вершинам графа G(V, E) всі ваги t_j, мінімальний час реалізації алгоритму A не може перевищити довгі (найдовші~~ого~~) шляхи M_{17} у графі G. Очевидно, що така обчислювальна структура ідеальна для даного подання графа алгоритму A, оскільки вона забезпечує ~~дозвіл~~ розв'язання задачі за мінімальний час при максимальному розпаралелюванні процесів обчислень. Ранг критичного шляху (кількість ребер у шляху) у графі $G(V, E)$, що відповідає деякому алгоритму A, будемо називати глибиною алгоритму A, а максимальну кількість алгоритмів $A_j \in A$, які можна одночасно виконати на довільному r-му кроці реалізації алгоритму A, – шириною алгоритму. Надалі глибину й ширину довільного алгоритму позначимо буквами a й h . Припустимо, що кожний з алгоритмів $\{A_j\} \in A$ реалізується на будь-якому процесорі $П_j$. Нехай задано структурну схему алгоритму A графом $G(V,E)$. Виникає задача визначення оптимальної стратегії розподілу алгоритмів $\{A_j\}$ по процесорах $\{П_j\}$, при якій час розв'язання задачі відрізняється від довжини критичного шляху в графі $G(V, E)$ на мінімально можливу величину., ~~О~~обрана стратегія їхнього використання повинна визначити архітектуру обчислювальної системи. Якщо ширина алгоритму дорівнює h, ~~,~~ то~~,~~ природно спробувати обмежитися h-процесорами для реалізації алгоритму A.

Для цього подамо граф $G(V, E)$ алгоритму А ув вигляді стягнутого дерева всіх шляхів. Дерево всіх шляхів D графа $G(V, E)$ (рис. 5.1, а), побудоване від першої вершини, як показано на рис. 5.1, б. Воно будується за матрицею суміжності $B = \| b_{ij} \|$ від деякої вершини Sтакий спосі.

Позначимо вершину S і беремо S-й рядок матриці В.

1. За цим рядком вибираємо номер вершин j_1, для яких $b_{sj} \neq 0$, і утворюємо множину вершин, що мають від вершини S шлях рангу $r = 1$ (вершини першого ярусу).

2. За цим рядком вибираємо номер вершин j_1, для яких $b_{sj} \neq 0$, і утворює множину вершин, що мають від вершини S шлях рангу $r = 1$ (вершини першого ярусу).

3. Вершини r-го ярусу зі шляхами рангу r від вершини S перебувають почерговим переглядом рядків вершин j^{r-1} в (r – 1) ярусі, які не зустрічалися в шляху від S до j^{r-1}.

4. Побудова дерева триває або до одержання шляхів максимального можливого рангу, або до тих пір, поки для вершин jr-1 не виявиться, що всі вершини вже ввійшли в шлях μs_j^{r-1} .

У загальному випадку кількість гілок у такому дереві $(N – 1)!$, тобто для утворення потрібно $(N – 1)!$ регістрів пам'яті. Щоб уникнути подібного роду труднощів, перейдемо від дерева D всіх шляхів до стягнутого дерева всіх шляхів (рис. 5.1, в). Воно може бути отримано стягуванням однойменних вершин дерева D на кожному ярусі. При цьому вершини з однаковими номерами на кожному ярусі впорядковуються в горизонтальні лінійки. Як видно з рис. 5.1, в, множина шляхів у стягнутому дереві шляхів таке саме, як і в дереві D (рис. 5.1, б), але для прочитання шляхів на кожній горизонтальній лінійці дозволяється бувати один раз.

Архітектура обчислювальної системи, що реалізує побудову дерева всіх шляхів довільного графа, наведена в роботах [154, 155]. Розглянемо стратегію розподілу процесорів по алгоритмах A_i у припущенні, що в нас є N-процесорів і кількість вершин у графі алгоритму так само дорівнює N.

Стратегія полягає в тім, щоб задіяти процесори у відповідності зі стягнутим деревом шляхів графа алгоритму A. Інакше кажучи, на першому кроці працює процесор $П_1$, реалізуючи алгоритм A_1, розташований на першому ярусі ($r = 1$). Далі за матрицею суміжності встановлюється зв'язок між процесорами $П_1$, $П_2$, $П_3$ і $П_4$. Після закінчення роботи всіх процесорів $П_2$, $П_3$, $П_4$ відповідно до матриці суміжності У встановлюється зв'язок процесорів $П_2$, $П_3$, $П_4$ із процесорами $П_5$, $П_6$. Результати роботи процесорів $П_2$, $П_3$, $П_4$ передаються на процесори $П_5$, $П_6$, далі процесори $П_5$,$П_6$ реалізують алгоритми A_5, A_6. Після закінчення їхнього виконання за матрицею суміжності встановлюється зв'язок між процесорами $П_5$, $П_6$ і процесором $П_7$ і йому передається вся необхідна інформація для роботи. У такий спосіб розв’язання задачі здійснюється в процесі побудови стягнутого дерева шляхів. На кожному етапі обчислень взаємодіють

процесори, що відповідають вершинам у стягнутому дереві шляхів, розташованим на n-му і $r + 1$ ярусі.

Ми розглянули приклад (рис. 5.1), коли структурна схема алгоритму складена так, що алгоритми A_j на кожному ярусі одержують необхідні вихідні дані для їхньої реалізації. У загальному випадку при довільному задаванні структурної схеми алгоритму ця умова може не дотримуватися, наприклад якщо структурна схема алгоритму A задана графом $G(V, E)$, поданим на рис. 5.2, а. На рис. 5.2, б наведена матриця суміжності графа G і значення ступеня d^- – результату кожної вершини й d^+ – ступеня заходу. Зрозуміло, що ступінь вершини графа G дорівнює $d = d^- + d^+$. У цьому випадку побудова стягнутого дерева шляхів (рис. 5.2, б) відрізняється лише тільки тим, що від вершини i зв'язок з вершинами наступного ярусу встановлюється лише коли ступінь заходу вершини досягла величини, обумовленої матрицею суміжності B. Фактично це означає, що процесор з номером j займається з першої подачі на нього інформації, необхідної для реалізації алгоритму A, і він залишається зайнятим доти, поки кількість звертань до нього не досягне ступеня заходу d_j^+. Ступінь вершини графа G дорівнює $d = d^- + d^+$. Це означає, що вся вихідна інформація для виконання алгоритму A_j отримана і процесор j здатний його реалізувати. Відповідно до стягнутого дерева шляхів (рис. 5.2, в) виконання алгоритму A, заданого графом G (рис. 5.2, а), здійснюється в такий спосіб. Процесор $П_1$ виконує алгоритм A_1, далі відповідно до B установлюються зв'язки з процесорами $П_2$, $П_4$, $П_7$, і на них пересилаються результати виконання A_1. При цьому процесор $П_2$ одержує всю необхідну інформацію й відпрацьовує алгоритм A_2 . Процесори $П_4$, $П_7$ зайняті й перебувають у режимі очікування. Далі за В/У встановлюються зв'язки з процесорами $П_3$, $П_4$, $П_6$ і передається інформація, необхідна для їхньої роботи. Ступінь заходу вершини 4 стане $d^+ = 2$, що обумовлено матрицею B і, отже, процесор $П_4$ може виконати алгоритм A_4. На рис. 5.2, в позначені зірочками (*) номери процесорів, що звільнилися на кожному ярусі.

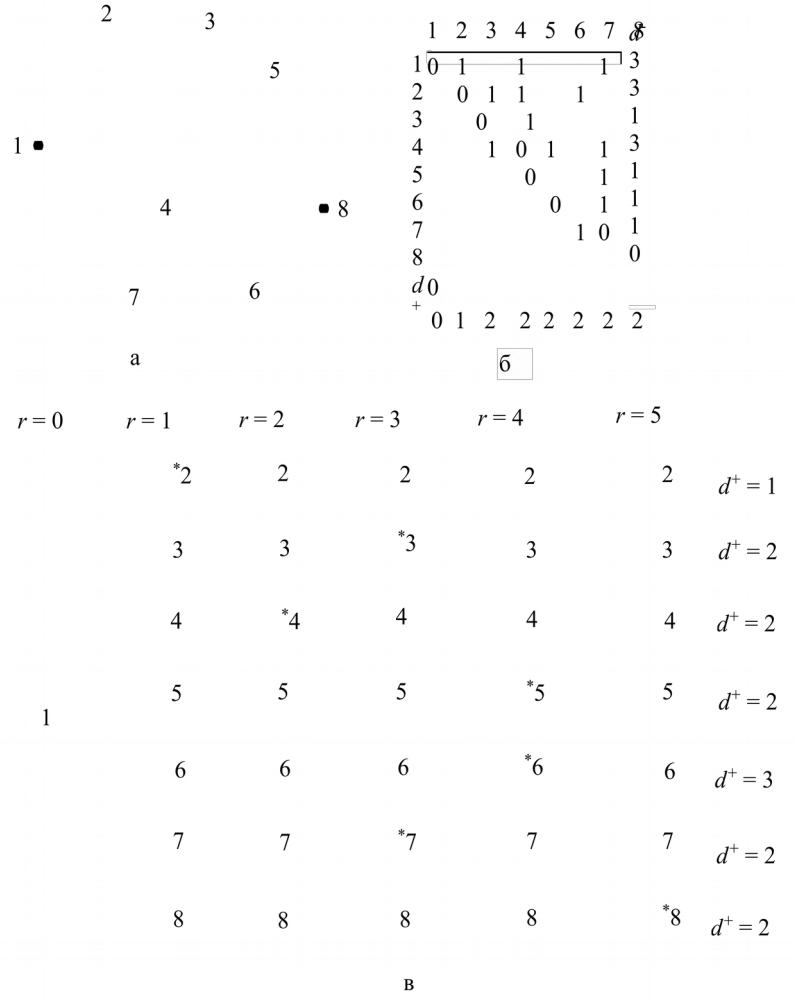


Рис. 5.2. Реалізація алгоритму на основі стягнутого дерева шляхів

Для процесора $П_4$ визначаємо за B зв'язки з процесорами $П_3$, $П_5$, $П_7$, при цьому процесори $П_3$ і $П_7$ виконують свої алгоритми й передають інформацію на процесори $П_5$, $П_6$, які реалізують алгоритми A_5, A_6, і передають вихідні дані на процесор $П_8$. Останній і завершить виконання алгоритму A.

Для здійснення такої організації обчислень передбачається використовувати обчислювальну структуру (рис. 5.3), яку ми й будемо називати циклічною паралельною обчислювальною системою. Вона містить n-процесорів (що складаються з пам'яті M_i і арифметичного логічного пристрою A_i), входи й виходи яких приєднані до комутатора розмірності (n*n). Комутатор управляється пристроєм управління введення й (ПУВВ) інформації відповідно до матриці B.

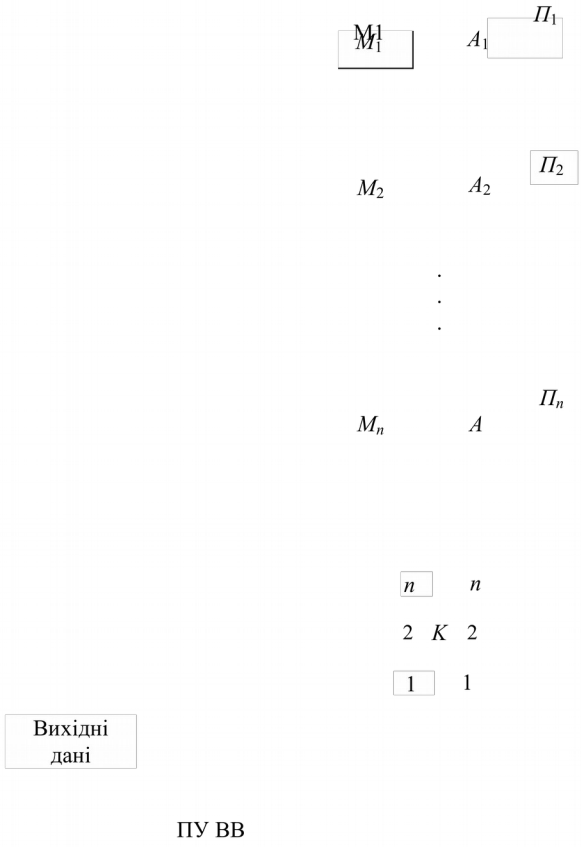


Рис. 5.3. ПОС циклічного типу

Фактично така структура обчислювальної системи й реалізує побудову стягнутого дерева шляхів графа, забезпечуючи зв'язок між процесорами при переході від ярусу до ярусу за допомогою ПУВВ і комутатора *K*, на основі матриці *B*. Обчислювальний процес у такій структурі можна подати у вигляді паралельно обертових кілець, рух яких є синхронним або асинхронним залежно від прийнятої організації синхронізації. Слід зазначити, що якщо кількість процесорів дорівнює *N* – кількості вершин у структурній схемі алгоритму, заданого графом *G*, то на кожному кроці роботи процесорів частина з них простоює. Якщо максимальна ширина алгоритму *A* дорівнює *h*, то одночасно на довільному ярусі буде працювати не більше *h*-процесорів.

Такою кількістю процесорів можна обмежитися в кільцевій структурі, за рахунок чого зменшити кількість процесорів, що простоюють. Цього можна досягнути, якщо при переході від ярусу до ярусу в дереві стягнутих шляхів будуть автоматично змінюватися номери процесорів, що комутуються, відповідно до матриці суміжності B. Архітектура кільцевої системи набуває вигляду, як на рис. 5.4, де ПУП – пристрій уравління перенумерацією комутатора для залучення процесорів, що відповідають наступному ярусу стягнутого дерева шляхів.

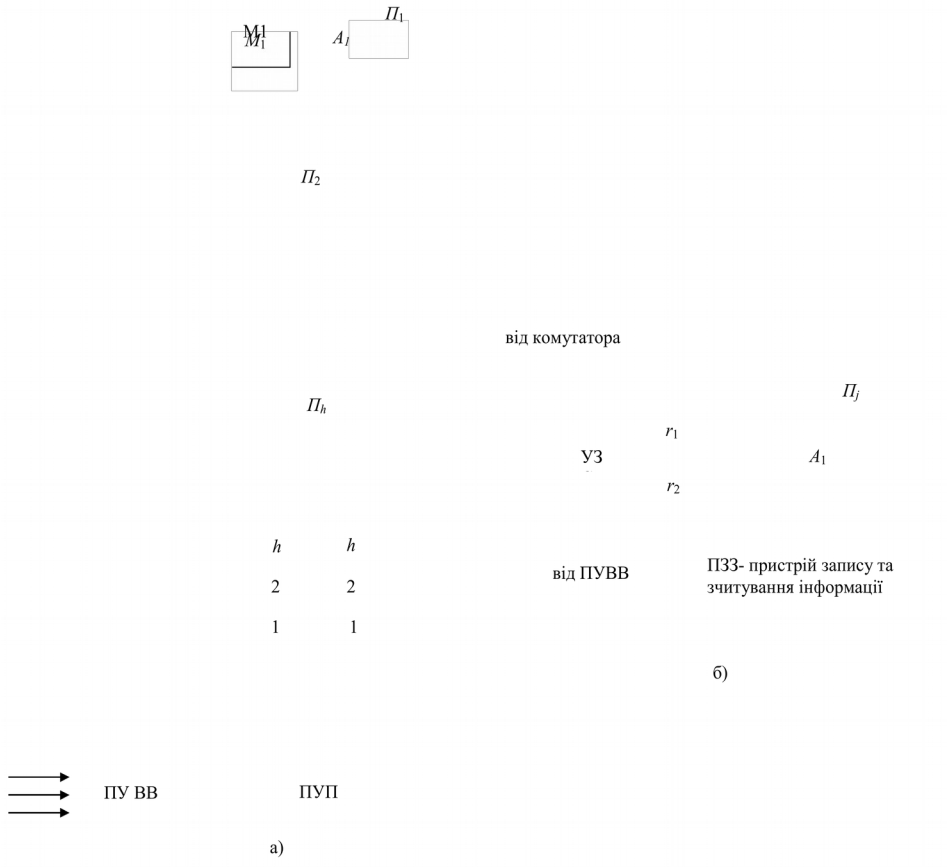


Рис. 5.4

При реалізації алгоритму A на кільцевій структурі час виконання алгоритму відрізняється від довжини критичного шляху тільки на величину at_k, де t_k – час спрацьовування комутатора. При побудові комутатора на основі програмувальних логічних матриць t_k визначається час перемикання логічного елемента "І". Якщо $t_k < \min\{t_j\}$, то час виконання алгоритму A на циклічній обчислювальній системі буде в точності дорівнювати довжині критичного шляху.

5.3. Варіанти побудови циклічних паралельних обчислювальних систем

Процес реалізації алгоритму A розбивається на множину алгоритмів $\{A_j\}$, кожний з яких у свою чергу знову розбивається на підмножину алгоритмів $\{A_j\}'$ і т. д. доти, поки алгоритм буде являти собою просто одну з базових операцій $y_i \in Y$. Множина Y утворить функціонально повну систему. Тому процесори $П_i$ доцільно створювати з використанням трансп’ютерної технології побудови високопродуктивних мікропроцесорів, утворених за принципом RISC–архітектури, що має локальну пам'ять, що являє собою два регістри й пристрої запису й зчитування інформації. RISC процесор містить арифметичний пристрій A_i (рис. 5.4, б), що забезпечує виконання базових арифметичних операцій. Процесор у цьому випадку

реалізує зазначені операції над двома числами й, отже, структурна схема алгоритму *A* повинна представлятися орієнтованим графом, у якому ступенів заходу вершини не більше двох, а ступеня результату - довільна. У процесорах можна звільнитися від ПЗЗ (рис. 5.4, б), тобто розпаралелити процес запису вихідної інформації в регістрі пам'яті, що вимагає додаткового комутатора *К*, що працює паралельно з основними. При цьому архітектура обчислювальної системи буде мати вигляд як на рис. 5.5.

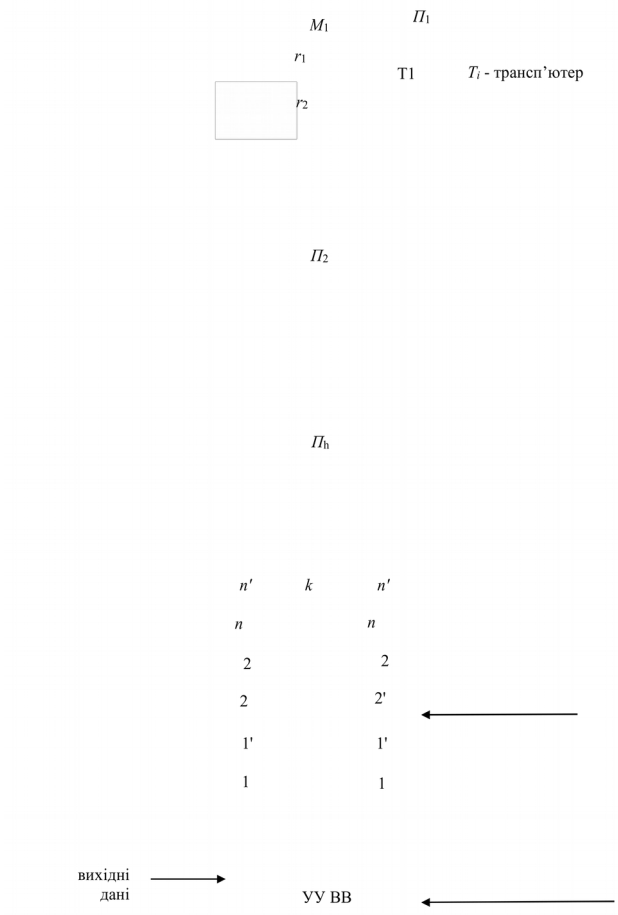


Рис. 5.5

Для оцінки роботи алгоритмів на таких системах використовуємо критерій – коефіцієнт прискорення

$$K_в = \frac{T_1}{T_n},$$

де T_1 – час реалізації алгоритму *A* на гіпотетичній однопроцесорній ЕОМ з такою самою швидкодією, як і $П_j$, і оперативною пам'яттю, що дорівнює сумарній пам'яті $П_j$, за наявності необхідної кількості зовнішніх пристроїв зі швидкостями обміну інформацією, як і в циклічній обчислювальній

системі;

T_n – час реалізації алгоритму А на циклічній обчислювальній системі, що містить n-процесорів.

Якщо алгоритм A представлений структурною схемою, заданої графом G, що містить n-вершин, то

$$T_1 = \sum_{i=1}^{n} t_i + nt_0, \tag{5.1}$$

де t_i – час виконання і – базової операції;

t_0 – час обміну.

Час роботи алгоритму A на структурах, зображених на рис. 5.5, можна визначити так:

$$T_n = \sum_{j=1}^{a} t_j^{\max} + at_0, \tag{5.2}$$

де $t_j^{\max}$ – найбільший із часів базових операцій, виконуваних одночасно декількома процесорами на j-му кроці;

a – глибина алгоритму A, заданого графом G.

Тоді коефіцієнт прискорення набуде вигляду:

$$\hat{E}_{ó} = \frac{\sum_{i=1}^{n} t_i + nt_0}{\sum_{j}^{a} t_j^{\max} + at_0}. \tag{5.3}$$

Для спрощення аналізу покладемо $t_i = \overline{t_k}$ – середнє значення по $i = (1, n)$, а $t_j^{\max} = \overline{t_a}$ – середнє значення по $j = (1, a)$, при цьому

$$\hat{E}_{ó} = \frac{n}{a}\alpha, \tag{5.4}$$

де $\alpha = \frac{\overline{t_k} \pm t_0}{\overline{t_a} \pm t_0}$.

Як видно з виразу (5.4), зі зменшенням глибини алгоритму A зростає коефіцієнт прискорення. У випадку, коли кожний процесор спеціалізується на виконанні алгоритмів деякої підмножини алгоритмів $\{A_j\}$, можлива організація обчислень циклічного типу на обчислювальній структурі, зображеній на рис. 5.6.

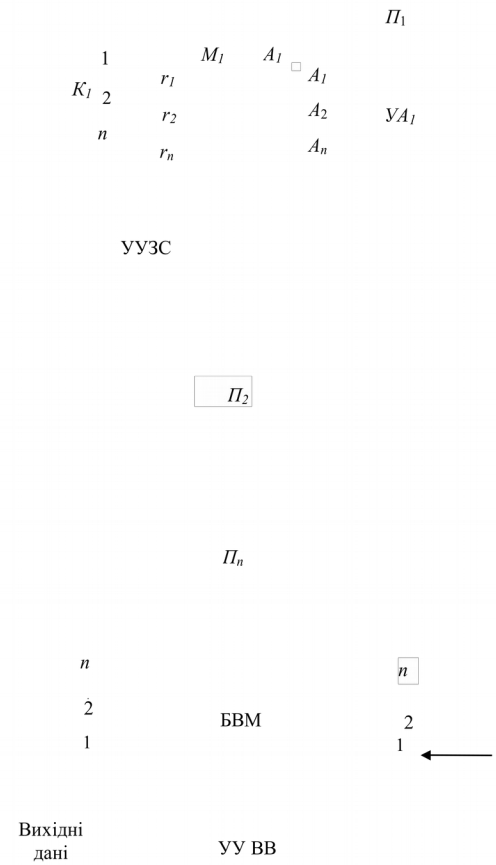


Рис. 5.6

У цій структурі пам'ять процесора $П_i$ включає (n – 1) регістр, закріплений за кожним $П_i$ процесором ($i = \overline{1,n}$); комутатор $К_i$; пристрій управління записом і зчитуванням; арифметико-логічні блоки (за кількістю регістрів); пристрій аналізу $УА_i$, що за певним критерієм вибирає з результатів роботи A_j найкращий. Кільця циркуляції інформації замикаються через блок вибору напрямку передачі інформації (БВН), він послідовно здійснює опитування процесорів $П_i$. Інформація процесора I=1 за допомогою комутатора $К_i$ записується в регістри пам'яті всіх процесорів $П_i$, закріплених за процесором з номером l.

Після закінчення опитування процесорів вони виконують операції за отриманими даними, а результати їхньої роботи знову за допомогою БВН і комутаторів заносяться в регістри пам'яті, закріпленій за процесорами.

Описані операції виконуються до повної реалізації алгоритму A. Для цього варіанта, коефіцієнт прискорення визначається виразом

$$\hat{E}_{ó} = \frac{\sum_{i=1}^{n} t_i + nt_0}{\sum_{j=1}^{a} t_j^{\max} + \alpha\, nt_0}. \qquad (5.5)$$

Щоб забезпечити прискорення, у цьому випадку повинна виконуватися нерівність

$$\sum_{i=1}^{n} t_i - \sum_{j=1}^{a} t_j^{\max} > nt_0(a\text{-}1) \qquad (5.6)$$

при $t_i = t_k$ і $t_j^{max} = t_0$,

$$\frac{n}{a} > \frac{t_k \pm nt_0}{t_a \pm t_0} . \qquad (5.7)$$

У реальних задачах час виконання алгоритму $A_j \in A$ різній. Тому якщо $\Delta t = \left(t_1^{\max} - t_2^{\min}\right)$ – час між закінченням виконання найскладнішого алгоритму $A_1 \in \{A_i\}$ і найпростішого $A_2 \in \{A_j\}$ використовуваних для опитування процесів $П_i$, то прискорення, що випливає з виразу (5.5), може зрости до величини, обумовленої співвідношенням (5.4). Останнє еквівалентно припущенню, що $n = 1$ у знаменнику дробу, обумовленого співвідношенням (5.5). Таку структуру кільцевої мережі ефективно застосовувати для визначення найкоротших шляхів, розв'язання задач динамічного програмування й варіаційного обчислення. Задачі варіаційного обчислення ефективно модулюються на основі підходів, викладених у роботі [127]. Реалізація кільцевих структур (рис. 5.4 і 5.6) можлива й на базі багатоканальних систем зв'язку, що використовують частотне або тимчасове ущільнення. Кількість каналів зв'язку буде визначатися кількістю регістрів пам'яті, наявних у процесорах $П_i$, тобто кожний процесор забезпечується багатоканальним приймачем і передавачем (рис. 5.7 і 5.8).

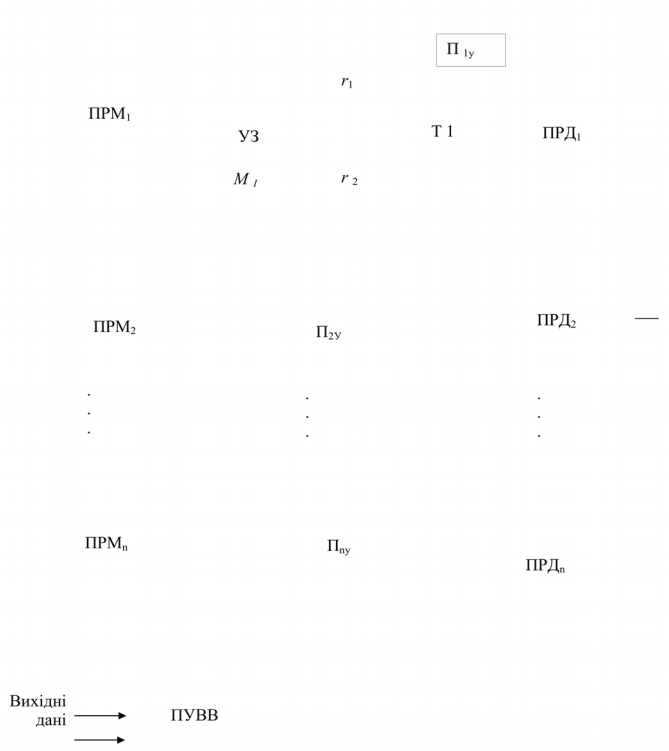


Рис. 5.7

Розглянуті структури циклічного типу зможуть успішно конкурувати з конвеєрними обчислювачами, побудованими на базі однорідних обчислювальних систем.

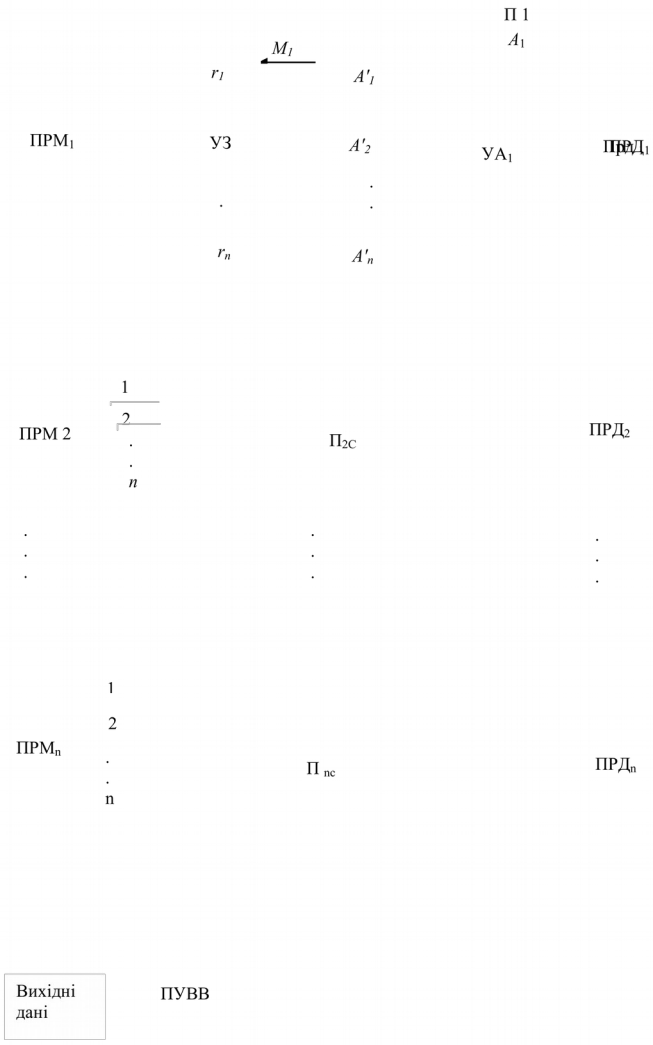


Рис. 5.8

Цю властивість продемонструємо на прикладі. Припустимо, що граф алгоритму має вигляд, показаний на рис. 5.9, а. Подібні графи властиві багатьом обчислювальним методам лінійної алгебри й математичної фізики [127]. У загальному випадку ґрати мають розмірність mxn. Однорідне обчислювальне середовище, що реалізує даний алгоритм, наведено на рис. 5.9, б. Його можна замінити на кільцеву обчислювальну структуру, що містить *h*-процесорів, зображену на рис. 5.4, а, де *h* – ширина аналізованого алгоритму. Швидкодія залишається практично такою самою, а виграш у кількості процесорів буде $\frac{m \times n}{h}$ раз.

У нашому конкретному випадку $m = n = h = 5$ і отже виграш в апаратурних витратах буде теж дорівнює 5. Підвищення продуктивності

циклічних паралельних обчислювальних систем залежить від їхньої вкладеності.

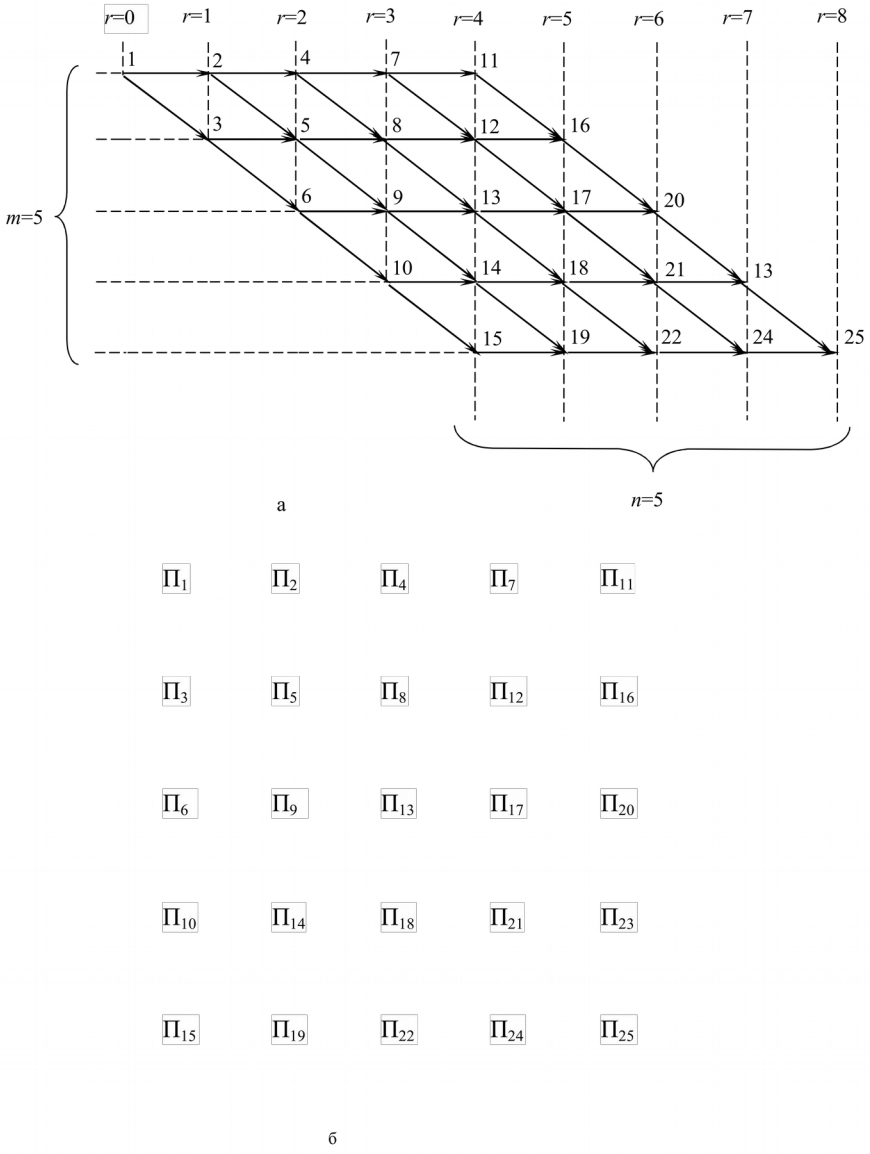


Рис. 5.9. Граф алгоритму (а) та однорідне обчислювальне середовище (б)

5.4. Вкладеність циклічних паралельних обчислювальних систем

Припустимо, що задано структурну схему алгоритму A у вигляді графа $G(V, E)$. Кожний алгоритм A_j, що відповідає i-вершині графа G, представимо графом $G_i(V_i, E_i)$. Кожний алгоритм A_j^i (граф G_i) у свою чергу можна представити у вигляді графа $G_{ij}(V_{ij}, E_{ij})$ і т. д. до тих пір, поки вершині не буде відповідати одна з базових елементарних операцій. У цьому випадку будь-який процесор $П_i$ для виконання алгоритму A_j буде обчислювальною структурою кільцевого типу (рис. 5.3), де кожний

процесор теж кільцева структура й т. д. Ступінь вкладеності таких систем довільний. На мікрорівні він обмежується кількістю функцій процесора, які не можуть бути менше якоїсь кількості базових операцій, необхідних для розв’язання поставлених задач, а також обумовлені технологічними можливостями виготовлення процесорів. У вкладеній системі процесор $П_i$ можна розглядати як "матрьошку", у якій містяться "процесори-матрьошки" меншої величини й т. д. У кожному процесорі $П_i$ пронумеруємо "процесори-матрьошки" від 1 до $У$, починаючи з найменших. Архітектура передбачуваної циклічної системи показана на рис. 5.10. Процес обчислень у такій системі відбувається так. Одночасно вмикаються процесори $П_1(1)$, $П_2(1)$, …, $П_n(1)$, результати своєї роботи вони передають процесорам $П_1(2)$, $П_2(2)$, …, $П_n(2)$ і т. д. доти, поки одночасно не запрацюють процесори $П_1$, $П_2$, …, $П_n$. По закінченні їхньої роботи задача буде розв’язана. За такою системою дуже ефективно розв’язуються однотипні задачі у випадку, коли їхня кількість досить велика. Після того, як процесори $П_1(1)$, $П_2(1)$, …, $П_n(1)$ закінчили працювати, вони звільняються й на них можна розв’язувати наступну задачу.

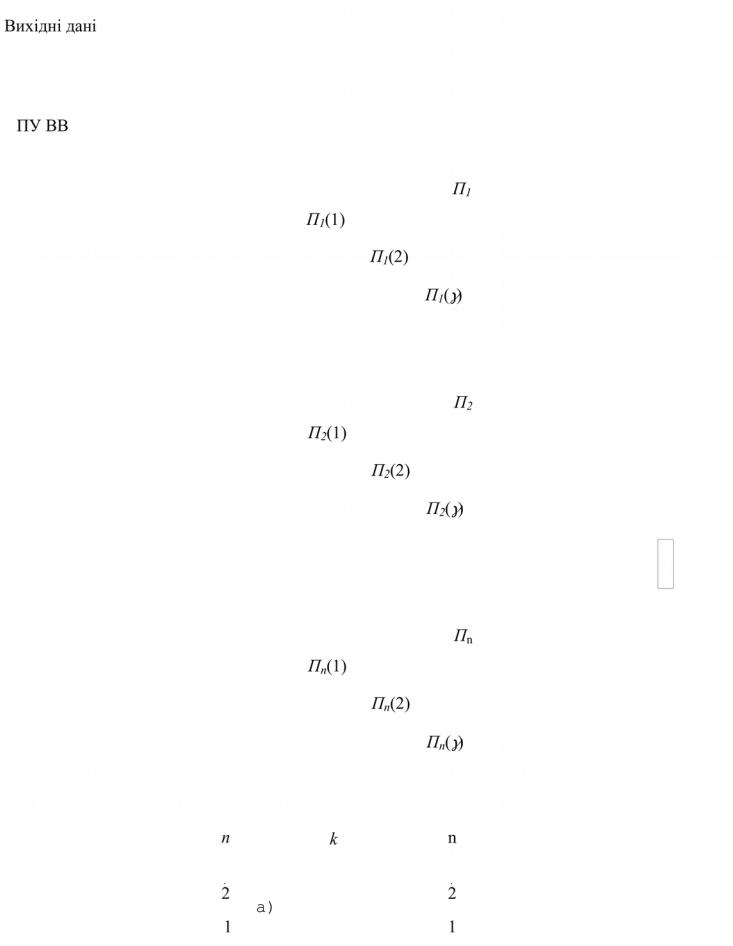


Рис.5.10

Якщо по черзі подавати задачі на розв’язання, то на процесорах $П_1$, $П_2$, …, $П_n$ буде закінчуватися розв’язання першої задачі, на процесорах $П_1(1)$, $П_2(1)$, …, $П_n(1)$ – уже почнеться розв’язання задачі γ. Отже, якщо ступінь вкладеності кожного процесора $П_i$ дорівнює γ, то на ~~ПВС~~ ПОС можна одночасно розв’язувати γ задач, причому всі "процесори-матрьошки" будуть увесь час завантажені.

Для розв’язання довільних різнотипних задач доцільно використовувати структуру циклічного типу, зображену на рис. 5.4, де процесори реалізовані за трансп’ютерною технологією й кількість пристроїв комутації й перенумерації номерів комутатора (УКП) мають бути за кількістю типів розв'язуваних задач. Процесори підключаються до УКП за допомогою складань "І", на які сигнали управління подаються з пристрою розподілу вільних на даний момент часу процесорів на УКП (рис. 5.11).

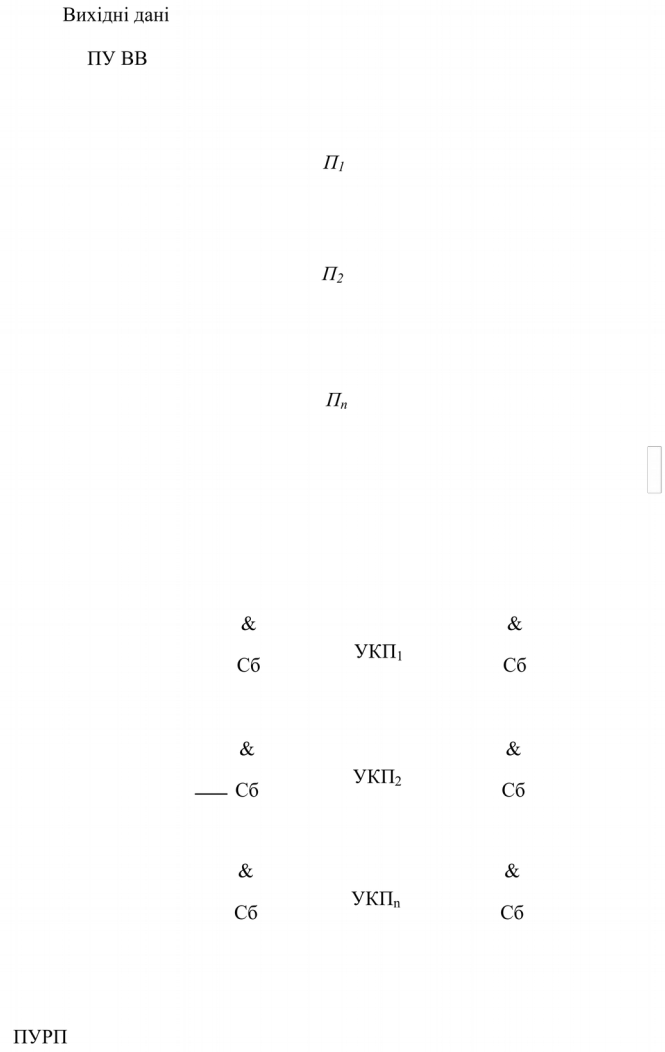


Рис. 5.11

При такій організації обчислень одночасно розв'язується *q*-різнотипних задач, які залежно від ширини алгоритмів їхнього розв'язання вимагають на кожному кроці підключення до УКП певної кількості процесорів. Тут можлива ситуація, коли на якомусь кроці деякі задачі зажадають всі наявності, що є в наявності, і тоді розв'язання задач, що залишилися, буде затримуватися. У більшості випадків комплекс розв'язуваних задач переслідує деяку кінцеву мету й очевидно, що вона буде швидше реалізована, якщо сумарний час розв'язання даного комплексу задач буде мінімально.

Розглянемо задачу визначення оптимальної стратегії розподілу процесорів $П_i$ по УКП ($i = 1 \div n$), при якій сумарний час T розв'язання всього комплексу задач мінімальний.

Позначимо через A^j_r комплекс алгоритмів, які необхідно виконати на r-му кроці роботи для розв'язання j-ї задачі ($i = 1 \div n$). У кожному комплексі алгоритмів є алгоритм із найбільшим часом реалізації. Такі алгоритми будемо називати критичними й позначати $A^j_r(кр)$. Тоді час виконання j-ї задачі дорівнює

$$t_{uj} = \sum_{r=1}^{a_j} t^j_r + \sum_{r=1}^{a_j} t\Pi_{ogr}(t_{r(i)}), \qquad (5.8)$$

де $x^j_r = x^j_{ir}$, якщо передбачається виділення $t\Pi_{ogr}(t_{r(i)}) = \begin{cases} 0 \\ \Delta t^j_r \end{cases}$ – в іншому випадку;

$Ï_y(t_i)$ – план розподілу процесорів $П_i$ по задачах у момент часу t_i;

Δt^j_r – час простою j-ї задачі на r-му кроці виконання при недостатній кількості процесорів $X^i_{rн}$;

t^j_r – час виконання A^j_r.

Мінімальний час розв'язання всіх задач із урахуванням виразу (5.8) визначиться в такий спосіб:

$$T = \min\left\{\max\left[\sum_{r=1}^{a_j} t^j_r + \sum_{r=1}^{a_j} t\Pi_{ogr}(t_{r(i)})\right]\right\}; \qquad (5.9)$$

$$X_j = \begin{cases} 1 - \text{якщо реалізується алгоритм } A^j_r; \\ 0 - \text{в іншому випадку;} \end{cases} \quad \sum_{j=1}^{q} b_j x_j \le b_c(t_i); \sum_j b_j \le n; \qquad (5.10)$$

де b_j – кількість процесорів, необхідних у момент часу t_j для виконання A^j_r;

$b_c(t_i)$- кількість вільних процесорів у момент часу t_i;

n – загальна кількість процесорів у системі (рис. 5.10).

Таким чином, оптимальний розподіл процесорів по задачах з метою мінімізації їхнього сумарного часу виконання T являє собою мінімаксну

задачу, у якій необхідно знайти таку послідовність планів: $П_2(t_1)$, $П_r(t_2)$, …, $П_r(t_j)$, коли буде досягатися мінімум функціонала (5.9). Задачу знаходження оптимального розподілу процесів по задачах у момент t_i можна сформулювати в такий спосіб: $\max \sum_{j=1}^{q} x_j$ при обмеженнях $\sum_{j=1}^{q} b_j x_j \le b_c(t_i)$; $0 \le b_j \le n, 0 \le \mathrm{b_j} \le \mathrm{n}$ $x_j \in \{0,1\}$. Ця задача являє собою задачу лінійного програмування з булевими змінними, відому як задача про рюкзак. У ній шукається максимальна кількість A_r^j, яку можна виконати в момент часу t_i вільними процесорами. Цей процес доводиться повторювати багаторазово при переходах від одного циклу обчислень до іншого.

У роботі [100] показано, що дана задача зводиться до задачі визначення екстремальних шляхів з обмеженням по вазі й може бути ефективно розв'язана в масштабі реального часу в мікросекундному діапазоні на обчислювальних структурах циклічного типу. Для вдосконалення структури необхідна спеціалізація процесорів на розв'язання різних класів задач. Процесори можуть мати більше глибоку вкладеність, тобто кожний процесор системи (рис. 5.11) у свою чергу являє собою обчислювальну систему (рис. 5.10). Тоді така система дозволить мінімізувати час виконання потоків задач різного типу.

5.5. Паралельні обчислювальні структури для розв'язання задач булевого лінійного й динамічного програмування

ПОС циклічного типу для реалізації рангового підходу

Для задачі ЦЛП із БЗ паралельна форма узагальненої процедури A_0 збігається з графом DΔ. У цьому випадку ~~ПВС~~ ПОС циклічного типу має вигляд, показаний на рис. 5.12, а. Як видно з рис. 5.12, а, роль комутатора виконує процесор $П_x$. Процесори $П_i$ i = (1, n) служать для зберігання й вибору шляху, що відповідає обраному правилу відсікань $\{L_w\}$. ПУВВ відповідає за початкове завантаження вихідних даних (ваг вершин, каліброваних векторів і т. д.) і виведенння результату після одержання припустимого або екстремального розв'язку. Всім процесом управляє пристрій управління (ПУ). Кожний $П_i$ процесор відповідає i-й вершині графа $D\Delta$ і зберігає постійно всі властиві їй ваги. На цій структурі задача розв'язується. Пристрій управління формує n раз управляючу послідовність тактів. У кожному такті формується множина шляхів у процесорі $П_x$, що відповідають номеру k цього такту. Для цього із процесорів $П_i$ і J k вибираються за правилами відсікання $\{L_w\}$ шляхи, які узагальнюються в $П_x$. Після цього результат сформованої множини записується в k-й процесор. І так доти, поки всі множини поточного рангу не будуть сформовані.

По закінченню формування шляхів рангу r виробляється побудова шляхів рангу $r = r + 1$ і так доти, поки або всі множини не будуть

порожніми (що відповідає нульовому лічильнику векторів поточного рангу), або ранг шляху стане рівним n. Слід зазначити, що для організації зберігання множин як попереднього, так і наступного рангів необхідно передбачити режим поділу адрес оперативної пам'яті усередині процесорів, щоб при записі множини сформованого рангу не затерти множину збережених векторів поточного рангу. На рис. 5.12, б зображено один з варіантів структури процесора $П_i$.

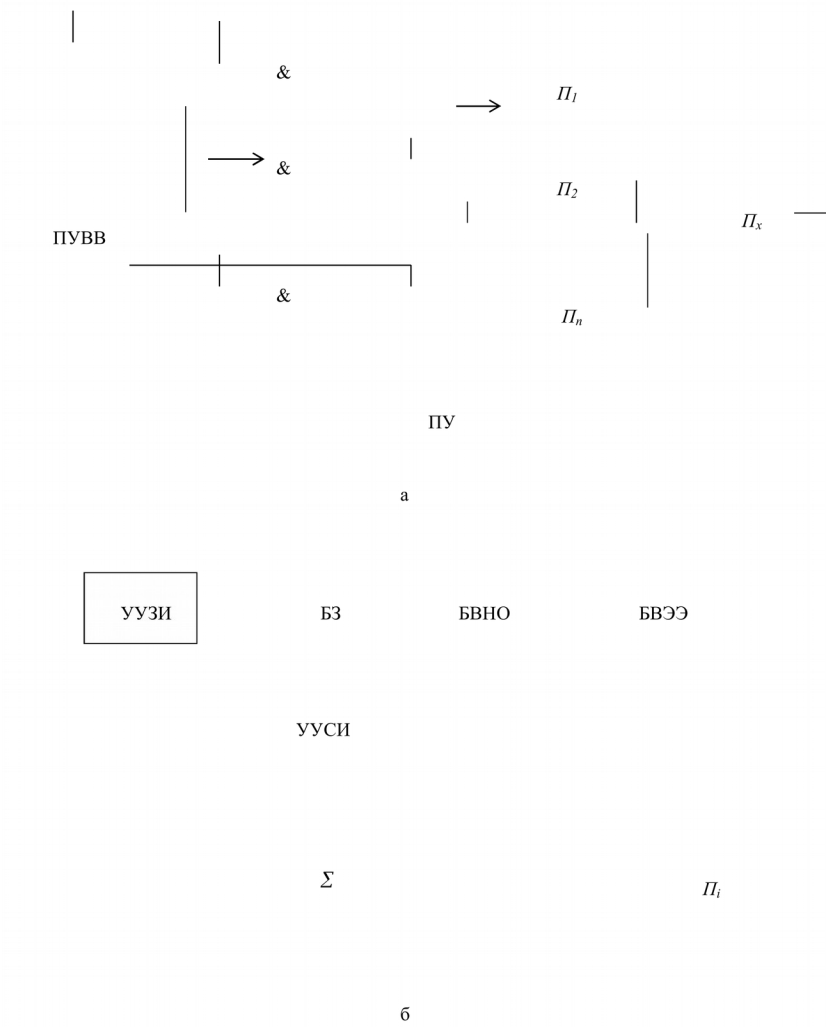


Рис. 5.12. ~~ПВС~~ ПОС для узагальненої процедури A_0

ПОС циклічного типу для розв'язання задач динамічного програмування

Як видно з проведеного аналізу, на паралельних обчислювальних системах циклічного типу можливе ефективне розв'язання варіаційних задач оптимального управління, диференціальних ігор, інтегральних рівнянь, а також оптимізаційних задач на графах, задач динамічного й цілочисленного програмування. При цьому використання ґрат одного типу дозволяє зробити структуру ~~ПВС~~ ПОС адаптивною до всіх перерахованих типів задач. Структура такого типу наведена на (рис. 5.12, а). Тут, як управляюча машина, може використовуватися будь-яка персональна ЕОМ. За допомогою схем «І» забезпечується одночасне уведення *m* чисел у

процесори $П_1 \ldots П_m$. За синхронізуючим сигналом із пристрою управління результати виконання операцій за m кроків передаються знову на входи процесорів П1 … Пm за допомогою Пх, при цьому на інші входи процесорів подаються наступні m ваг ребер, необхідні для виконання наступного кроку визначення екстремальних шляхів на ґратах. Ваги ребер ґрати є вихідними даними для розв'язання задачі. Для багатьох задач дискретної оптимізації, таких як задача синтезу мінімальної вартості, задачі про призначення, задачі лінійного програмування з булевими змінними, вони відразу задаються у вигляді матриць ваг, і на підготовку даних до розв'язання цих типів задач не потрібно додаткового часу. У випадку розв'язання задач дискретної оптимізації на основі зведення їх до задачі визначення екстремальних шляхів потрібен попередній розрахунок ваг ребер ґрати. У роботах [76, 78] розглянуто підходи зведення задач динамічного програмування до задачі визначення екстремальних шляхів на ґратах такого типу як на рис. 5.9, а, з яких видно, що цей процес досить простий, і в більшості випадків можна здійснювати підготовку даних у процесі розв'язання задачі або заздалегідь, залежно від вимог, пропонованих до розв'язання.

Аналогічно при розв'язанні варіаційних задач: тут ваги ґрати являють собою певні інтеграли, які можна, знаючи функціонал й межі інтегрування, обчислити заздалегідь до розв'язання задачі, але можна розрахувати і на базі фоторецепторних полів (пристрій уведення-виведення), застосовуваних для інтегрування в машині «Мозаїка» [127, 169]. При цьому кожний цикл роботи обчислювальної системи збільшиться на час інтегрування *t* фоторецепторного поля, а тривалість розв'язання всієї задачі, якщо ґрати довжини *n*, зросте в *nt*, тобто не змінить порядку тимчасової складності алгоритму розв'язання задачі. Більш докладні можливості технічної реалізації запропонованої структури наведено в роботах [157-162].

Використання ~~ПВС~~ ПОС систолічного й хвильового типу для розв'язання задач булевого програмування

Як уже було зазначено раніше, однією з переваг рангового підходу до розв'язання задач ЦЛП є можливість його реалізації на ~~ПВС.~~ ПОС. Оскільки розрахунок у вершинах графа *D*Δ на кожному ранзі здійснюється незалежно один від одного, то якщо в якості операторної вершини графа *D*Δ уявити процесорний елемент, можна показати можливість реалізації описаних у роботі алгоритмів на паралельних архітектурах. Це дозволить в *n* раз зменшити витрати на розв'язання задач БЗ.

У роботі показана можливість реалізації розроблених алгоритмів на циклічних ~~ПВС~~ ПОС систолічного й хвильового типу. Вони можуть бути використані як прискорювач, приєднаний до сумісної провідної ЕОМ, або як самостійна ЕОМ, оснащена управляючим процесором.

Головна ЕОМ повинна забезпечувати системне управління й управління даними, установлювати програму, що управляє всіма пристроями системи, і генерувати коди глобального управління. Вона може бути обрана з широкого діапазону машин, що охоплює різні рівні обчислювальної потужності. Блок управління процесором являє собою програмувальний процесор (контролер), що служить інтерфейсом між спеціальною високошвидкісною шиною й спеціальним обчислювачем. Разом з управлінням шинним протоколом він генерує адреси для доступу до буферної пам'яті й видає тимчасові управляючі сигнали для спеціального обчислювача. Схема поєднання спеціального обчислювача з головною ЕОМ показана на рис. 5.13. Систолічні масиви добре пристосовані для СБІС-реалізації обчислювально-ємних алгоритмів. Вони мають такі переваги, як модульність, регулярність, локальні міжз'єднання, високий ступінь конвеєрної мультиобробки й безперервний потік даних між ПЕ. Недолік систолічних масивів полягає в глобальному управлінні їхнім функціонуванням за допомогою потактової синхронізації.

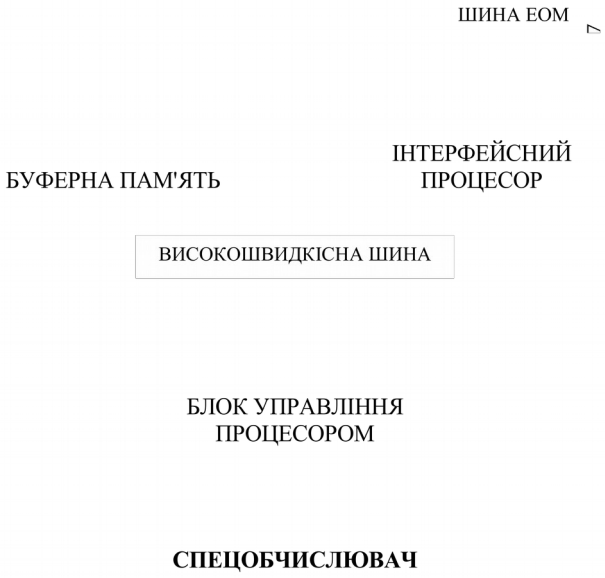


Рис. 5.13. З’єднання спеціального обчислювача з ЕОМ

В апаратурній реалізації синхронізація призводить до проблем розфазування синхронізуючих імпульсів, відмовостійкості й пікової потужності. Для спеціалізованої МШС реального часу операційна система не потрібна, тому що система виконує одну функцію тривалий час і будь-яку зміну функції можна виконати в автономному режимі.

Циклічна ~~ПВС~~ ПОС систолічного типу складається з 4 пристроїв (рис. 5.14): обчислювального пристрою 1; обчислювального пристрою 2;

модуля пам'яті; блока управління систолічним процесором. Обчислювальний пристрій (ЛЕ) 1 ($ОУ_1$) призначено для виконання обчислень локальних екстремумів при заданому функціоналі й обмеженні, а також визначення (обчислення) номера вершини, у якій локальний екстремум визначений. Пристрій складається з n – 1 процесорних систолічних комірок, реалізованих на основі простої структури, і характеризується однорідністю й локальністю зв'язків. Кожна процесорна комурка має зв'язок тільки з сусідньою коміркою. Систолічні комірки працюють паралельно, після виконання обчислень відбувається обмін даними між сусідніми процесорними комірками. Кожна процесорна комурка складається з блока регістрів (БР), призначеного для зберігання й забезпечення мікрооперації передачі інформації між регістрами БР сусідніх процесорних комірок; арифметичного обчислювача (АВ), використовуваного для обчислення ЛЕ на підставі даних, що надходять із БР, вибору ЛЕ за максимальним значенням функціонала й мінімальним значенням обмеження й пересилання його в блок визначення глобального екстремуму $ВУ_2$ для обчислення глобального екстремуму; блока ідентифікації (БІ), службовця для визначення номера вершини, у якій ЛЕ визначений. Обчислювальний пристрій 2 призначено для обчислення з вступників ЛЕ глобального й формування вектора шляху. Пристрій складається з двох блоків: блока визначення глобального екстремуму (БВГЕ); блока визначення вектора шляху (БОВШ). Блок визначення глобального екстремуму служить для обчислення глобального екстремуму й складається зі схеми порівняння, регістрів локального й глобального екстремумів.

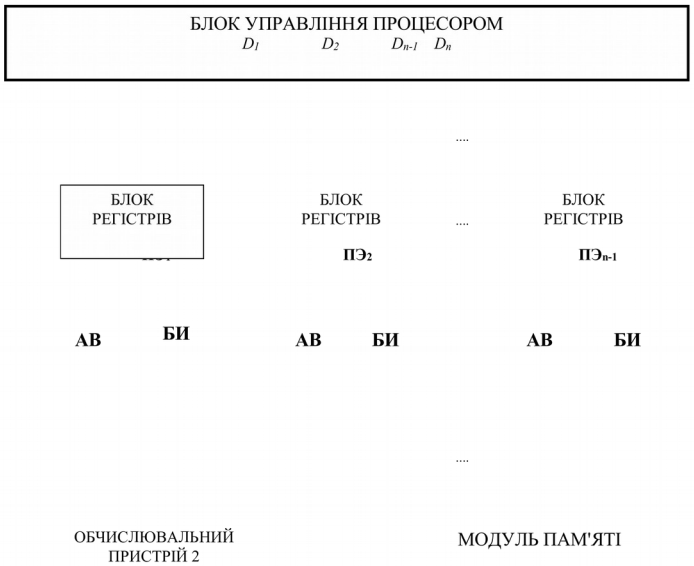


Рис. 5.14. Архітектура ~~ПВС~~ ПОС систолічного типу

Блок визначення вектора шляху використовується для обчислення номера останньої вершини визначального вектора шляху й складається з

трьох суматорів, чотирьох лічильників, двох тригерів, регістрів адреси й проміжних обчислень. Модуль пам'яті призначений для зберігання номерів вершин локальних екстремумів на кожному ранзі обчислень. Складається з дешифратора адреси й пам'яті. Для виконання обчислень дані надходять одночасно в кожну систолічну комірку.

Уведення даних здійснюється під управлінням блока управління систолічним процесом з буферної пам'яті. Ця пам'ять використовується як буфер між високошвидкісною спеціалізованою шиною й низькошвидкісною шиною ЕОМ. За необхідності дані в буфері оновляються, зчитуються й обробляються систолічною матрицею. Завантаженням буферної пам'яті управляє інтерфейсний процесор, функціональне призначення якого полягає в упралінні взаємодією ЕОМ і інтерфейсної системи, плануванні й перевірці обчислень, виконуваних систолічною матрицею. У табл. 5.1 показано порядок обчислень на ~~ПВС~~ ПОС систолічного типу.

Таблиця 5.1

Порядок обчислень у ~~ПВС~~ ПОС систолічного типу

Крок	Виконання обчислень				
	$ПЕ_1$	$ПЕ_2$	. . .	$ПЕ_{n-1}$	$ПЕ_n$
1	1-2	2-3	. . .	(n-2)-(n-1)	(n-1)-n
2	-	1-3	. . .	(n-3)-(n-1)	(n-2)-n
. . .	-	-	. . .	. . .	. . .
n-1	-	-	-	1-1-(n-1)	2-n
n	-	-	-	-	1-n
Ранг 2	$ЛЕ_1$	$ЛЕ_2$	. . .	$ЛЕ_{n-1}$	$ЛЕ_n$
n+1	ЛЕ 1-_3	ЛЕ 2-_4	. . .	$ЛЕ_{n-1-n}$	-
n+2	-	ЛЕ 1-_4	. . .	$ЛЕ_{n-2-n}$	-
. . .	-	-	. . .	. . .	-
2 *n*-1	-	-	-	ЛЕ 1-_n	-
Ранг 3	$ЛЕ_1$	$ЛЕ_2$	. . .	$ЛЕ_{n-1}$	-
. . .	. . .	. . .	. . .	-	-
n(*n*-1)-2	ЛЕ 1-_1-(n-1)	ЛЕ 2-_n	-	-	-
n(*n*-1)-1		ЛЕ 1-_n	-	-	-
Ранг *n*-1	$ЛЕ_1$	$ЛЕ_2$	-	-	-
n(*n*-1)	ЛЕ 1-_n	-	-	-	-
Ранг n	$ЛЕ_1$	-	-	-	-

На першому кроці (такті) у $ПЕ_1$ буде отриманий $ЛЕ_1$, на другому – $ЛЕ_2$ у $ПЕ_2$ і т. д. Таким чином, для обчислення глобального екстремуму

необхідно витратити $N^*(N-1)/2$ тактів, що на порядок нижче порівняно з обчисленнями в однопроцесорному режимі.

Хвильовий матричний процесор, показаний на рис. 5.15, являє собою обчислювальну мережу, що має такі властивості: автосинхронне обчислення, керовані даними; регулярність, модульність, локальні міжз'єднання; можливість програмування хвильовою мовою.

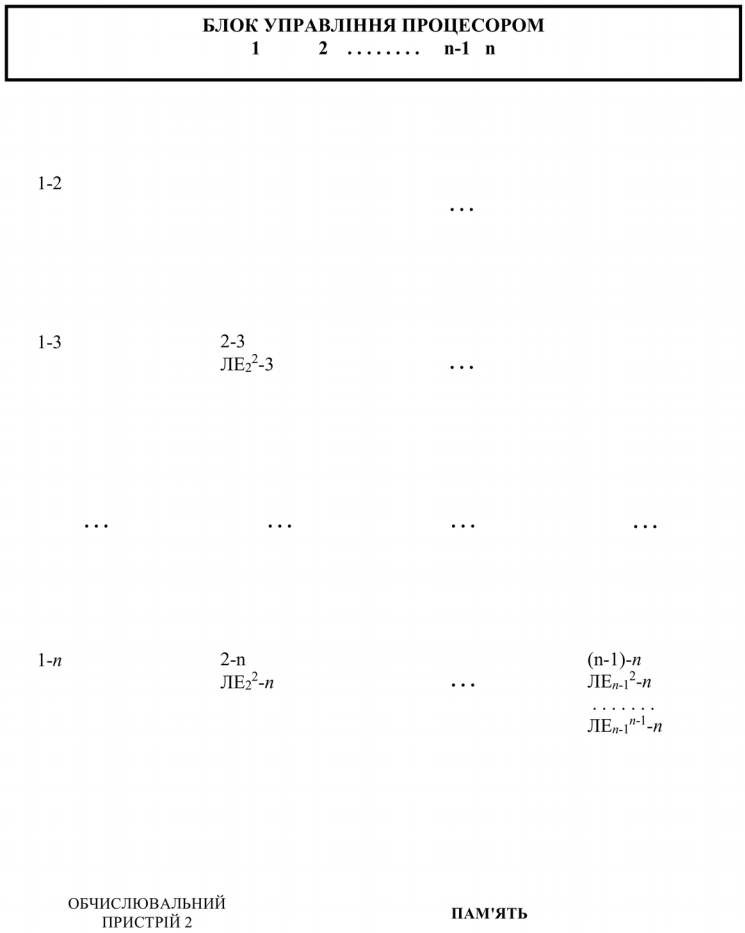


Рис. 5.15. Порядок обчислень на хвильовій ПОС

Архітектура ~~ПВС~~ ПОС хвильового типу або шляхом задавання графа потоку даних; конвеєр з лінійним зростанням прискорення. Цей процесор має більшість переваг систолічного процесора й, крім того, робить обчислення, керовані потоком даних. Порядок обчислень на хвильовий ~~ПВС~~ ПОС показано на рис. 5.15. На першому кроці будуються всі шляхи рангу 2, на другому кроці з локальних екстремумів 2 рангу $ЛЕ_i^{r=2}$ будуються шляхи рангу 3, на третьому кроці з локальних екстремумів 3 рангу $ЛЕ_i^{r=3}$ будуються шляхи рангу 4 і т. д. доти, поки не буде знайдено остаточний розв’язок. Таким чином, на ~~ПВС~~ ПОС матричного типу задача може бути розв’язане не більш ніж за N - 1 тактів.

5.6. Визначення оптимальних маршрутів і шляхів з максимальною пропускною здатністю в телекомунікаційних мережах на основі ідей рангового підходу

5.6.1. Класифікація й аналіз існуючих методів розв'язання задач про найкоротший шлях

Будемо розглядати граф, ребрам якого привласнені деякі ваги. Такі графи називають зваженими графами. Ваги ребер позначаються l_{ij}, де j – суміжні вершини, які задаються матрицею довжин L = l_i. Якщо $\mu_{st} = (s, v_1, v_2, \ldots, v_k, t)$ – деякий шлях з s в t, то довжина цього шляху визначається як сума довжин ребер, утворюючих даний шлях. Поточну величину довжин між вершинами i, j позначимо $d(i,j)$, а через P(i,j) позначимо величину найкоротшого шляху між парою вершин i і j.

Розглянемо відомі підходи до розв'язання задачі визначення найкоротших шляхів. У загальному випадку при розв'язанні цієї задачі можливі варіанти, коли граф, до якого зводиться вихідна задача, має як додатні, так і від'ємні ваги ребер. У літературі виділяють такі випадки: у графі всі ребра з додатними вагами, у графі є ребра з від'ємними вагами, але немає циклів від'ємної довжини; у графі є цикли від'ємної довжини. При розв'язанні даних задач використовуються такі методи: індексні або позначок; матричні; апаратні.

Всі ці методи використовуються для розв'язання двох видів задач: при знаходженні найкоротшого шляху між заданою парою вершин і найкоротших шляхів між всіма парами вершин у графі.

Для розв'язання задачі пошуку найкоротшого шляху між парою вершин кращими є індексні й апаратні методи. Для розв'язання задач визначення найкоротших шляхів між усіма парами вершин у графі кращими є матричні методи. Одним з найвідоміших алгоритмів визначення найкоротших шляхів у графі з додатними вагами ребер є алгоритм Дейкстри. Його використання в графах з від'ємними вагами ребер призводить до експонентного зростання його складності [78]. У випадку, коли в графі є цикли від'ємної довжини, задача належить до класу NP-повних задач.

Навіть перерахування всіх алгоритмів визначення найкоротших шляхів не є можливим, тому що кількість публікацій з методів визначення найкоротших шляхів зараз уже перевищило 200 [178]. Свою увагу зупинимо на найбільш відомих індексних (Дейкстри, Форда-Белмана) і матричних алгоритмах (Флойда-Уоршелла, Мерчланца, Шимбела); апаратних методах (аналогових, цифрових).

5.7. Задача визначення найкоротшого шляху на основі рангового підходу

Як показано в роботі [178], дана задача може розглядатися і як задача лінійного програмування на основі одиничного потоку мінімальної вартості. У термінах математичного програмування дана задача формулюється в такий спосіб.

Мінімізувати

$$\sum_{i,j} l_{ij} X_{ij} \tag{5.11}$$

за умов

$$\sum_i X_{ij} - \sum_i X_{ji} = \begin{cases} 1, & \text{якщо } i = s; \\ -1, & \text{якщо } i = t; \\ 0, & \text{якщо } i \neq s, t; \end{cases} \tag{5.12}$$

$$X_{ij} \in \{0,1\}. \tag{5.13}$$

Треба визначити потік $\{X_{ij}\}$, мінімізуючий функціонал (4.1). Змінні X_{ij} є булевими зі значеннями $\{0,1\}$. Якщо ребро (i, j) належить найкоротшому шляху, то $X_{ij} = 1$, в іншому випадку $X_{ij} = 0$. Значення функціонала визначає $P(s,t)$. Обмеження (5.12), (5.13) відображають умови безперервності шляху. У даній роботі задача визначення найкоротшого шляху розв'язується за допомогою запропонованого рангового методу, в основі якого лежить перехід від графа G до стягнутого дерева всіх шляхів D_0.

Розглянемо деякий довільний граф $G(\mathrm{V}, \mathrm{E})$ із множиною вершин V і множиною ребер E, потужність якого не може перевищити $n(n-1)/2$. Якщо виділити в графі вільні вершини s і t, то множина всіх шляхів $\{\mu_{st}\}$ між даною парою вершин можливо подати як об'єднання підмножин

$$M_{st} = M^{r=1}_{st} \cup M^{r=2}_{st} \cup \ldots \cup M_{st}^{r=n-1}, \tag{5.14}$$

де M_{st} – множини шляхів між вершинами s і t рангу r.

Шляхи рангу $r = n-1$ є шляхами з максимально можливим рангом. Вони відомі в теорії графів як гамільтонові шляхи, які проходять через всі його вершини. Таким чином, задача визначення найкоротшого шляху між заданою парою вершин $s - t$ може бути сформульована як задача визначення шляху деякого рангу r мінімальної довжини в множині M_{st}:

$$\min_r M_{st} = \min \{M_{st}^{r=1}, M_{st}^{r=2}, \cdots, M_{st}^{r=n-1}\}. \tag{5.15}$$

З виразу (5.15) випливає, що якщо в кожній з підмножин M^r_{st} (r = 1, n–1) визначити найкоротший шлях рангу r і відповідно позначити його M^{*r}_{st} (r = 1, n –1), тоді задача визначення найкоротшого шляху зведеться до визначення найкоротшого шляху в множині $\{M^{*r}_{st}\}$, тобто

$$\min_r \{M^{*r=1}_{st}, M^{*r=2}_{st}, \cdots, M^{*r=n-1}_{st}\}. \qquad (5.16)$$

Припустимо, що деяка процедура A дозволяє будувати найкоротші шляхи між вершинами s і t вільного рангу, тоді відповідно до виразу (5.15), користуючись співвідношенням (5.16), визначаємо найкоротший шлях між вершинами s і t. У випадку визначення найкоротших шляхів між вершиною s і всіма іншими вершинами графа G задача може бути визначена так:

Визначити

$$\min \mu^r_{si} = \min\{\mu^{r=1}_{si}, \mu^{r=2}_{si}, \ldots, \mu^{r=n-1}_{si}\},$$

де $i = (1, n)$, $r = (1, n-1)$,
або якщо

$$\mu^r_{si} = \min\{M^{*r}_{si}\}, \text{ то } \min \mu^r_{si} = \min\{\mu^{r=1}_{si}, \mu^{r=2}_{si}, \ldots, \mu^{r=n-1}_{si}\}\ldots$$

Перехід від дерева всіх шляхів графа до стягнутого дерева всіх шляхів D_0 відповідає розбиттю множини шляхів графа G на підмножини M^{*r}_{si}, при цьому вершини графа D_0 можна розглядати як множини шляхів M^{*r}_{si}.

5.8. Алгоритми визначення найкоротших шляхів на основі рангового підходу

Останнім часом для розв’язання задачі знаходження найкоротшого шляху в графі стали використовувати однорідні багатопроцесорні обчислювальні структури [51-53, 169]. У зв'язку із цим виникла проблема створення високопродуктивних паралельних алгоритмів і структур для визначення найкоротших шляхів.

5.8.1. Алгоритми визначення найкоротших шляхів на основі процедури А

Побудова найкоротших шляхів ґрунтується на наступній процедурі A. З вершини s будуються всі можливі шляхи другого рангу до всіх вершин графа G. Отримана множина шляхів M^2_i розбивається на підмножини M^2_i так, щоб кожний з них закінчувався на i. У кожній підмножині виділяється

шлях мінімальної довжини. У результаті даної операції одержуємо $M^2_{\min}$ – множина шляхів другого рангу, у якому кожний шлях є найкоротшим шляхом другого рангу від вершини *s* до відповідної вершини *t*. Використовуючи множину $M^2_{\min}$, будуються всі шляхи рангу 3 до усіх вершин графа. Потім отримана множина M^3 розбивається на підмножини M^3_i, у кожній з яких визначається шлях мінімальної довжини.

Виконання процедури *A* повторюється до одержання шляхів максимально можливого рангу $r = n - 1$. Очевидно, що після $(n - 1)$ кроку повторення процедури *A* будуть певні всі можливі кандидати на найкоротші шляхи між вершинами s і t рангів $r = 1, 2, ..., (n-1)$. Вибираючи серед них шляхи мінімальної довжини для кожної вершини, визначаємо найкоротші шляхи між вершиною *S/s* та іншими вершинами графа.

Таким чином, процес визначення найкоротших шляхів між заданою вершиною *S/s* та іншими вершинами графа складається з двох кроків.

Алгоритм A1

Крок 1. $(n-1)$ кратне повторення запропонованої процедури *A*, у результаті чого одержуємо множину мінімальних шляхів всіх можливих рангів до всіх вершин.

Крок 2. Вибір мінімальних шляхів до вершини *i* у кожній з отриманих на кроці 1 множин шляхів, які і є шуканими найкоротшими шляхами від вершини *S/s* до інших вершин графа *G*.

Теорема 5.1. Алгоритм *A1* дозволяє визначити найкоротші шляхи між заданою вершиною *S* у графі *G* та іншими його вершинами.

Доведення. Нехай процедура побудувала всі шляхи рангу 1 від вершини S/s до інших вершин графа й на їхній базі – всі можливі шляхи рангу $r^{**} = 2$, тобто сформовано всі підмножини M^{r}_{si}. Виберемо в кожній підмножині найкоротший шлях $\mu^{r=2}_{i}$. Зрозуміло, що шляхи $\mu^{r=2}_{i}$ є найкоротшими шляхами рангу $r = 2$ до вершин *i*, оскільки на цьому кроці кожна з підмножин містить у собі всі можливі шляхи рангу $r = 2$. Сформуємо тепер відповідно до роботи алгоритму A_1 на базі шляхів $\{\mu^{r=2}_{si}\}$ всі можливі шляхи рангу $r = 3$, тобто множини $M_i^{r=3}$, і виділимо в кожному найкоротший шлях $\mu^{r=3}_{si}$. Доведемо, що отримані в такий спосіб шляхи $\mu^{r=3}_{si}$ є найкоротшими шляхами рангу $r=3$ з вершини *S* у вершини $i = (1, n)$, $i \neq s$. Для цього припустимо, що до деякої вершини $i = j$ є шлях $\mu^{r=3}_{si}$, довжина якого менше, ніж $\mu^{r=3}_{si}$. Оскільки довжини шляхів рангу r=3 відповідно до запропонованої процедури визначалися як

$$d_{si}^{r=3} = \min \{ d_{si}^{r=2} + l_{ij} \}; \; i = (1, n); j = (1, n); \; i \neq j,$$

це припущення може бути правильним тільки у випадку, якщо в графі є

більш короткий шлях $\mu_{si}^{**r=2}$ рангу r=2, ніж шляхи $\mu_{si}^{*r=2}$, що неможливо, тому що вони є найкоротшими шляхами рангу $r = 2$ у графі G.

Тепер припустимо, що алгоритм A_1 побудував найкоротші шляхи $\mu_{si}^{*r=k}$ рангу $r = k$ до всіх вершин графа. Далі відповідно до A_1 побудуємо на їхній основі всі можливі шляхи рангу $r = k + 1$, тобто сформуємо множини $M_i^{r=k+1}$ і в кожній множині $M_i^{r=k+1}$ (i = 1, n) виділимо найкоротші шляхи $\mu_{si}^{*r=k+1}$ і доведемо, що вони є найкоротшими шляхами рангу r=k+1. Для цього повторимо попереднє міркування, тобто припустимо, що є шлях $\mu_{si}^{**r=k+1}$ коротше, ніж шляхи $\mu_{si}^{*r=k+1}$, що суперечить первісному припущенню про те, що побудований алгоритмом А1 шлях $\mu_{si}^{*r=k}$ є найкоротшим шляхом рангу $r = k$ і отже шляхи $\mu_{si}^{*r=k+1}$ дійсно є найкоротшими шляхами рангу $r = k + 1$.

У такий спосіб ми показали, що алгоритм A_1 при r = 3 визначає найкоротші шляхи рангу r = 3. Далі припустили, що він утворив найкоротші шляхи рангу $r = k$, і довели, що на їхній основі він побудував найкоротші шляхи рангу $r = k + 1$, отже відповідно до методу повної математичної індукції алгоритм A_1 дозволяє визначати найкоротші шляхи довільного рангу.

Після виконання (n-1) кроку у всіх множинах M_i^r будуть побудовані шляхи μ_{si}^{*r} і тоді відповідно до співвідношення (5.16) визначаємо найкоротші шляхи між вершиною S/s і всіма іншими вершинами графа, що й було потрібно довести.

У процесі роботи алгоритму A_1 для довільних графів можливе виникнення ситуації, коли $M^r{}_i$ у процесі роботи алгоритму можуть виявитися порожніми. Це можливо у двох випадках: коли шляху поточного рангу r до вершини i у графі G немає й коли шлях є, але алгоритм A_1 не зміг його побудувати, – таку ситуацію будемо називати тупиковою. Покажемо, що виникнення тупикових ситуацій не збільшує оптимальності роботи алгоритму А1. Для цього розглянемо більш докладно механізм виникнення тупиків. Тупикова ситуація може мати місце в тому випадку, якщо $M_{it}^{r=q+1} = \varnothing$. Відповідно до алгоритму A_1 підмножина сформованих шляхів $M^r{}_i$ може бути порожньою, якщо вершина t увійшла в усі найкоротші шляхи рангу $r = q$ до вершин I = (1, n). Сформулюємо наступну теорему.

Теорема 5.2. Якщо в процесі роботи алгоритму А1 побудовано найкоротші шляхи наступних рангів $\mu_{st}^{*r=1}$, $\mu_{st}^{*r=2}$, … , $\mu_{st}^{*r=q}$ і підмножина шляхів $M_{st}^{r=q+1} = \varnothing$, то найкоротшим шляхом між вершинами S і t є шлях

$$\mu_{st} = \min \{ \mu_{st}^{*r=1}, \mu_{st}^{*r=2}, \dots, \mu_{st}^{*r=q} \} \dots$$

Доведення. Оскільки підмножина $M_{st}^{r=q+1} = \varnothing$, тоді вершина t увійшла в шляху $\mu_{si}^{*r=q}$, що являють собою найкоротші шляхи рангу $r = q$ з вершини S до всіх інших вершин графа $i = (1, n)$.

Припустимо, що є шлях $\mu_{st}^{**k=q+1}$ коротший, ніж $\mu_{st}^{*r=q+1}$. Це можливо, якщо існують шляхи $\mu_{st}^{**r=q}$ коротші, ніж шляхи $\mu_{st}^{*r=q}$, рангу $r = q$, що не проходять через вершину t, але це суперечить первісному припущенню про те, що шляхи $\mu_{st}^{*r=q}$, які проходять через вершину t, є найкоротшими шляхами рангу r = q до вершин i, отже, наше припущення про наявність шляху $\mu_{st}^{**r=q+1}$ коротшого, ніж μ_{st}^{*}, не є правильним, що й було потрібно довести.

Справедливість *теореми 5.2* випливає з властивості, всіх найкоротших шляхів – будь-який відрізок найкоротшого шляху також є найкоротшим шляхом між з'єднаними ним вершинами. З *теореми 5.2* випливає важливий висновок: якщо ваги в матриці довжин $L = \| l_{ij} \|$ є додатними величинами, то виникнення тупикових ситуацій не впливає на оптимальність роботи алгоритму A_1, оскільки найкоротші шляхи є джерелом виникнення тупиків. Значно складніше питання у випадку, коли в матриці ваг (довжин) містяться ваги від'ємної величини і є цикли від'ємної довжини, в аналізованому графі в цьому випадку оптимальність роботи алгоритму A_1 може порушуватися. У цьому випадку ми переходимо до іншого класу задач, відомих у теорії дискретної оптимізації як NP-повні задачі, можливість розв'язання яких на графі G має потребу в окремому обстеженні.

Фактично запропонований алгоритм А1 відображає основне функціональне рівняння динамічного програмування для даної задачі:

$$d_{\min}^{r}[j] = \min_{r} \{d_{\min}^{r-1}[i] + l_{ij}\}, \qquad (5.17)$$

де $d^{r-1}{}_{\min}[i]$ – мінімальна довжина $(r - 1)$ рангу від вершини *S/s* до вершини i.

Рівняння (5.17) дозволяє по ланцюжку знаходити умовні оптимальні розв'язки.

Оскільки запропонований алгоритм розгортає процес динамічного програмування від першого кроку до останнього, то знаходження оптимального розв'язку можливо після знаходження всіх умовних локальних розв'зків і відповідних значень локальних мінімумів. При цьому величини найкоротших шляхів від кореневої вершини *S/s* до всіх інших вершин і визначаються таким співвідношенням:

$$d_{\min}[i] = \min_{r} \{d_{\min}^{r-1}[i]\}. \qquad (5.18)$$

Використання розглянутого алгоритму відповідно до принципу оптимальності Беллмана [72, 73] дозволяє визначати найкоротші шляхи з деякої довільно взятої вершини *S/s* графа *G* до всіх інших вершин цього графа.

Відмінною рисою алгоритму A_1 є те, що на кожному кроці роботи порівнюються величини шляхів того самого рангу. Це, відрізняючись від алгоритмів, заснованих на методі розміщень міток [163], дозволяє зробити ефективне розпаралелювання алгоритму визначення найкоротших шляхів у графі.

5.8.2. Паралельний варіант алгоритму А1 (алгоритм А2)

Оскільки формування множин шляхів M на кожному ранзі можна робити одночасно, відповідно до співвідношень (5.15) і (5.16) для закінчення роботи алгоритму A_1 при його паралельній реалізації необхідно $(n - 1)$ раз виконати операції додавання й порівняння й одну операцію вибору мінімального елемента в масиві, обумовлену співвідношенням (5.16).

Після побудови множини шляхів M^{r+1} у множинах попереднього рангу M^r необхідно зберегти тільки екстремальні значення d^r_{min}, отже об'єм пам'яті, необхідної для реалізації алгоритму A_2, визначається множиною шляхів, сформованих на даному ранзі. Кількість таких шляхів дорівнює $n^2 - 3n + 2$, тому що в кожній підмножині M^r_i міститься $(n - 2)$ шляхів рангу r і таких підмножин $(n-1)$. Складність алгоритму A_2 не перевищує $0(n)$, об'єм пам'яті – $0(n^2)$. Складність алгоритму A_2 у послідовному варіанті його реалізації не перевищить $0(n^3)$. Слід зазначити, що швидкодію алгоритму А2 може бути підвищено, якщо в процесі виконання рівняння (5.17) ввести додаткову перевірку

$$d^r_{\min}[i] \le d^{r+1}{}_{\min}[\mathrm{i}] \qquad (5.19)$$

у кожній з формуючих множин M^r. Якщо на ранзі r співвідношення (5.19) виконується для всіх підмножин, то далі використовувати співвідношення (5.19) нема рації, тому що всі отримані на наступних кроках шляхи більш високого рангу будуть свідомо довше, ніж найкоротші шляхи, отримані до рангу $r + 1$. У найгіршому разі, коли найкоротшим шляхом є гамільтонів шлях, складність, звичайно, залишається $0(n)$. Остання ситуація зустрічається досить рідко, і введення перевірки нерівності (5.19) дозволить значно зменшити кількість кроків в A_2. Якщо визначається найкоротший шлях у графі рангу r, то за алгоритмом вони будуть знайдені на $(r + 1)$ кроці. Необхідна пам'ять для реалізації запропонованого алгоритму залишається пропорційною $0(n^2)$.

Приклад роботи алгоритму А2 розглянемо для мережі, зображеної на рис. 5.16, показано у табл. 5.2. Для мережі, заданої графом на рис. 5.16, від

вершини 1 потрібно побудувати дерево найкоротших шляхів до всіх інших вершин графа.

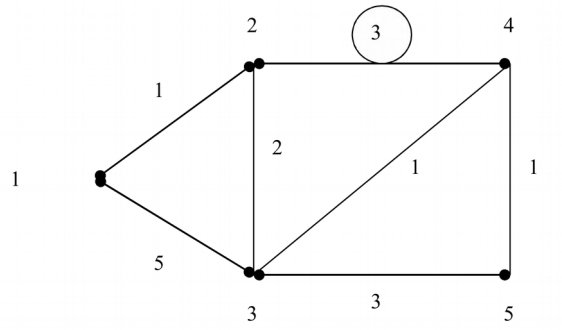


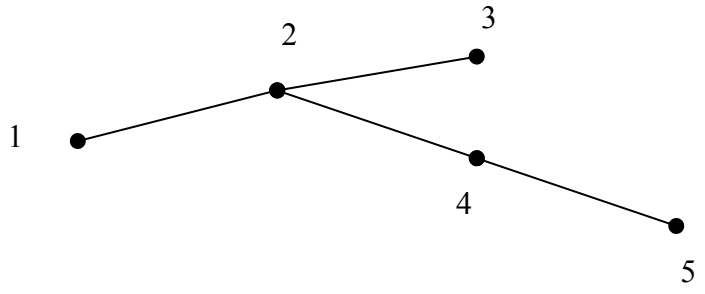
Рис. 5.16. Граф G

Таблиця 5.2

Процес побудови дерева найкоротших шляхів

12(1)* 2	132(7)* 2	----- 2	----- 2
13(5)* 3	123(3)* 3	1243(5)* 3	1245(8)* 3
---- 4	124(4)* 134(6) 4	1324(10) 1234(4)* 1354(9) 4	------ 4
---- 5	135(8)* 5	1235(6) 1245(5)* 5	12435(8) 12345(5)* 5
r=1	r=2	r=3	r=4

Вибираючи в кожній горизонтальній лінійці найкоротший шлях, ми одержимо дерево найкоротших шляхів



5.8.3. Паралельна реалізація алгоритму Дейкстри

Одним з недоліків алгоритму A_2 є те, що його практична реалізація ускладнюється тим, що для побудови на його основі паралельних архітектур обчислювальних систем у процесорних елементах необхідно мати n-1 групу регістрів для проміжного зберігання шляхів, при цьому, якщо мережа задається неповним графом, більша частина регістрів не використовується. З погляду трансп’ютерних технологій це істотно обмежує розмірність розв'язуваних задач при управлінні складними системами. У даній роботі пропонується алгоритм, що має таку саму

тимчасову складність, але дозволяє істотно спростити структуру процесорного елемента спеціального обчислювача. Припустимо, що відомо найкоротші шляхи $\overset{*r}{\mu}_{sj}$ в кожній підмножині m_{sj}^{r}. Тоді найкоротший шлях від вершини *S* до вершини *j* дорівнює $\overset{**}{\mu}_{sj} \min\limits_{r} \{\overset{*r=1}{\mu}_{sj}, \overset{*r=2}{\mu}_{sj}, ..., \overset{*r=n-1}{\mu}_{sj}\}$. Позначимо довжину шляху $\overset{*r}{\mu}_{sj}$ як $d(\overset{*r}{\mu}_{sj})$ і назвемо її локальним екстремумом рангу *r* до вершини *j*, $d(\overset{**r}{\mu}_{sj})$ – глобальним екстремумом рангу r до вершини *j*, а довжину $d(\overset{*r}{\mu}_{sj}) = \min\limits_{j} \{\overset{*r}{\mu}_{sj}\}$ – глобальним екстремумом на ярусі. Тоді задача визначення КЗ може бути розглянута як задача визначення глобальних екстремумів у просторі, обумовленим стягнутим деревом шляхів. Використовуючи властивість, що полягає в тім, що будь-який відрізок КЗ є теж КЗ між вершинами, що визначають даний відрізок, побудуємо наступну дуже просту процедуру визначення КЗ між заданою вершиною S і всіма іншими вершинами довільного графа *G*(*V, E*).

Алгоритм A_3

Крок 1. З вершини *S* будуємо $m_{sj}^{r=1}$ $j = \overline{(1, n-1)}$, виділяємо в них локальні екстремуми, серед них виділяємо глобальний екстремум на ярусі $d(\overset{*r=1}{\mu}_{sj})$ і вершину j виключаємо з подальшого аналізу.

Крок 2. На основі шляху, що відповідає поточному глобальному екстремуму на ярусі, будуємо всі можливі шляхи в множини $m_{sj}^{r:=r+1}$ і підтягуємо в них шляхи, що відповідають локальним екстремумам попереднього рангу. Після цього визначаємо локальні екстремуми в множинах $m_{sj}^{r:=r+1}$ і серед них виділяємо глобальний екстремум на ярусі $r = r + 1$, що відповідає деякій вершині j, і вершину j виключаємо з подальшого аналізу.

Крок 3. Перевіряємо $r = n - 1$, якщо ні, то переходимо до виконання кроку 2, інакше алгоритм закінчує роботу.

Пояснимо роботу даної процедури на прикладі. Нехай потрібно визначити КЗ у графі *G*, наведеному на рис. 5.16, від вершини 1 до всіх інших вершин графа. Процес роботи процедури на всіх її кроках роботи показаний у табл. 5.3, у якій кожний квадрат відповідає вершині в стягнутому дереві шляхів. У таблиці показано шляхи, побудовані процедурою на всіх ярусах. При цьому у таблиці зірочкою (*) позначено шляхи, що відповідають локальним екстремумам, а двома зірочками (**) – позначено шляхи, що відповідають глобальним екстремумам на ярусах. У дужках біля кожного шляху зазначена його довжина. Множини, у яких побудувати шляхи не вдалося, тобто порожні множини, у табл. 5.3

відзначені прочерками, а більш темним кольором у лінійках виділені вершини, що виключаються з подальшого аналізу. За табл. 5.3 визначено шляхи, що відповідають глобальним екстремумам у горизонтальних лінійках і їхньої довжини.

Таблиця 5.3

Процес побудови дерева найкоротших шляхів

12(1)** 2	---------- 2	---------- 2	---------- 2	12(1)
13(5)* 3	123(2)** 13(5) 3	---------- 3	---------- 3	123(2)
------- 4	124(4)* 4	1234(4) 124(4)** 4	---------- 4	124(4)
-------- 5	---------- 5	1235(6)* 5	1245(5)** 1235(6) 5	1245(5)
$r = 1$	$r = 2$	$r = 3$	$r = 4$	

Як видно з табл. 5.3, на першому кроці роботи процедури були побудовані шляхи рангу r = 1:12(1) з довжиною, що дорівнює 1, і шлях 13(5) довжиною 5, у множинах 4 і 5 побудувати жодного шляху не можна й, отже, вони залишилися порожніми. Тепер на першому ярусі виділяємо локальний екстремум, йому відповідає шлях 12(1); оскільки він на ярусі найкоротший, то вершину 2 виключаємо з подальшого аналізу, у таблиці вона зафарбована більш темним кольором. Далі на основі шляху 12 формуємо всі можливі шляхи на наступному ярусі, це шляхи 123(2) і 124(4), після цього підтягуємо локальні екстремуми попереднього рангу, це шлях 13(5), і після цього виділяємо глобальний екстремум на ярусі, йому відповідає шлях 123(2), а вершину 3 виключаємо з подальшого аналізу. На основі шляху 123(2) формуємо всі можливі шляхи на наступному ярусі, це шляхи 1234(4) і 1235(6). Далі підтягуємо шляхи, що відповідають локальним екстремумам попереднього ярусу, таким є тільки один шлях 124(4), і після цього на ярусі 3 виділяємо шлях, що відповідає глобальному екстремуму на ярусі таких шляхів 2: 1234(4) і 124(4), вибираємо з них кожної, наприклад 124(4), і вершина 4 виключається з подальшого аналізу. Далі на основі шляху 124(4) будуємо всі можливі шляхи на наступному 4-му ярусі, це тільки один шлях 1245(5). Підтягуємо шлях, що відповідає локальному екстремуму, з попереднього ярусу. Це шлях 1235(6), виділяємо шляхи, що відповідають локальним екстремумам, ним буде тільки один шлях 1245(5), підтягуємо шляхи, що відповідають локальним екстремумам попереднього ярусу, це тільки один шлях 1235(6), і після цього виділяємо

глобальний екстремум на ярусі, йому відповідає шлях 1245(5). Вершина 5 виключається з подальшого аналізу. Оскільки вже всі вершини виключені з аналізу, алгоритм закінчує роботу. Далі, виділяючи в кожній лінійці шляхи, що відповідають глобальним екстремумам, ми одержуємо КЗ від вершини 1 до всіх інших вершин графа. Ними є шляхи 12(1); 123(2); 124(4); 1245(5). Як видно з розробленої процедури й табл. 5.3, на кожному кроці роботи алгоритму в підмножинах не може міститися більше двох шляхів, у той час як в алгоритмах запропонованих у роботах [133, 141], кількість шляхів у підмножинах дорівнює (n-1), що істотно спрощує апаратну реалізацію процесорних елементів спеціального обчислювача. Оскільки операції з формування шляхів у кожній множині на ярусі можуть виконуватися одночасно, то на n процесорних елементах з урахуванням того, що на кожному ярусі доводиться робити одну операцію, пов'язану з виділенням мінімальної кількості з масиву, що не перевищує n чисел, загальна складність алгоритму не перевищить $O(n \log n)$. Слід зазначити, що якщо розпаралелити операцію вибору мінімального елемента в масиві за рахунок використання програмувальних логічних матриць, тимчасова складність алгоритму може бути знижена до $O(n)$. Важливою перевагою розробленого алгоритму є те, що він може бути реалізований на паралельних обчислювальних системах циклічного типу (див. розд. 6). Як показано в роботах [178, 204], час розв’язання задачі визначення найкоротших шляхів на спеціалізованій машині «Мозаїка», розробленій в інституті проблем моделювання в енергетиці, для визначення найкоротших шляхів у графі, що містить 256 вершин з середньою вагою ребра, що дорівнює 10, становить у середньому 68* 10-4з, на ЄС-1020 воно склало 1 хв 9 с, а на запропонованій структурі 25,6 = 10-5 с. Слід зазначити, що коефіцієнт прискорення для алгоритму А2 становить $n^2/\log_2 n$, а для алгоритму А3 $n/\log_2 n$, однак в алгоритмі А2 максимальна кількість шляхів, сформованих у множинах $m_{sj}^{r=r}$ на довільних рангах, не перевищує (n – 1), а для алгоритму А3 – двох шляхів. Але при цьому алгоритм A_2 завжди повинен будувати шляхи рангу $r = n - 1$, а алгоритм A_3 до деякого значення r_k, обумовленого виконанням нерівності (4.9). Тому становить інтерес оцінити середнє значення рангу найкоротших шляхів у довільних графах з ваговими характеристиками, що змінюються за рівномірним законом розподілу. Досліджувався ряд графів із кількістю вершин від 25 до 70 із щільністю ребер від 0,125 до 1, при цьому попередні дані отримані з довірчою ймовірністю 0,95, а наступні значення r_k відповідно до табл. 5.4.

Для порівняння в табл. 5.4 показано максимальний ранг дерева найкоротших шляхів. Отже коефіцієнт виграшу K_1 алгоритму A_2 порівняно з алгоритмом A_3 буде визначатися як n/r_k.

Таблиця 5.4

Значення середнього рангу найкоротших шляхів

n	25	30	35	40	45	50	60	70
r_k	7	8	8	8	9	9	9	10
max r	11	11	11	12	12	12	14	12

5.9. Визначення шляхів з максимальною пропускною здатністю в телекомунікаційних мережах на основі рангового підходу

При розв'язанні задачі маршрутизації повідомлень в обчислювальних мережах необхідно вміти визначати шляхи з максимальною пропускною здатністю, по яких можна передавати інформацію з мінімальними затримками. Оскільки сам процес маршрутизації повідомлень в обчислювальних мережах повинен здійснюватися в реальному масштабі часу, то до алгоритмів визначення шляхів з максимальною пропускною здатністю висуваються більш тверді тимчасові вимоги. Вони фактично повинні дозволяти розв'язувати цю задачу в прискореному масштабі часу. Розглянемо можливість розв'язання цієї задачі.

Постановка задачі на дослідження. Нехай у графі $G(V, E)$, що описує аналізовану мережу, кожне ребро (i, j) має пропускну здатність c_{ij} і задається матрицею пропускних здатностей $C = [c_{ij}]$. У графі задається пара вершин s, t. Потрібно визначити шлях μ_{st} від s до t з найбільшою пропускною здатністю $C^*(\mu_{st})$. Відомі алгоритми розв'язання цієї задачі [161] засновані на наступній теоремі.

Теорема 5.3. Пропускна здатність шляху з найбільшою пропускною здатністю від s до t дорівнює

$$\min_k \left[\max_{(i,j)\in k} \{ c_{ij} \} \right],$$

де k – будь-який s-t розріз у множині ребер E.

Під s-t розрізом у графі G розуміється мінімальна множина ребер графа G, вилучення яких з G перекриває всі шляхи з вершини s у вершину t [161]. Дана теорема подібна з теоремою про максимальний потік і мінімальний розріз, вона є мінімаксним варіантом останньої. Найбільш відомий алгоритм [161], заснований на цій теоремі, складається з таких кроків.

Крок 1. Почати з s-t розрізу $\overline{K}(\{s\}, V - \{s\})$ і знайти найбільшу пропускну здатність $\bar{c}$ с ребер з $\overline{K}$.

Крок 2. Побудувати остовий підграф $G(V, A)$, де $A = \{(i,j) \mid c_{ij} \geq C\}$.

Крок 3. Знайти множину вершин і $R(s)$, досяжних з s у графі $G(V, A)$.

Крок 4. Якщо $t \in R(s)$, то $c^* = \overline{C}$ і будь-який *s-t* розріз в остовому підграфі $G(V, A)$ буде мати найбільшу пропускну здатність c^*. Якщо $t \in R(s)$, перейти до наступного кроку.

Крок 5. Взяти в якості $\overline{K}$ розріз $(R(s), V - R(s))$ і знайти найбільшу пропускну здатність *C* ребер у цьому новому розрізі. Повернутися до кроку 5.

Для неорієнтованих графів Френка і Фриша, як показано в роботі [161], запропонували ще більш просту процедуру визначення шляху з найбільшою пропускною здатністю. Нехай K – деякий *s-t* розріз, а $\overline{c}$ - найбільша пропускна здатність ребер з K_1. Якщо тепер «закоротити» будь-яке ребро (i, j) із пропускною здатністю $c_{ij} \geq \overline{c}$, тобто вершини v_i, v_j замінити однією вершиною v, припустивши

$$\Gamma(v) = \Gamma(v_i) \cup \Gamma(v_j),\ \Gamma^{-1}(\mathrm{v}) = \Gamma^{-1}(v_i) \cup \Gamma^{-1}(v_j),$$

а ребро (i, j) видалити, то граф G_1, що отримали, буде мати той самий шлях з найбільшою пропускною здатністю, що й первісний граф *G*. Процедуру, застосовану до *G*, повторюють для графа G_1, вибираючи інший *s-t* розріз K_2, «закорочуючи» ребра з G_1, пропускна здатність яких не менше найбільшої пропускної здатності будь-якого ребра з K_2. Це дає граф G_2 і т. д. Процес «закорочування» ребер закінчується тоді, коли будуть «закорочені» *s* і *t*. Основним недоліком такого підходу є необхідність виділяти s-t розрізи, що може бути непростою задачею, якщо граф непланарний. Крім того, алгоритм Френка й Фриша застосуємо тільки для неорієнтованих графів, що визначається процесом «закорочування» ребер, тому що при цьому неявно передбачається, що шлях із пропускною здатністю не менше $\overline{c}$ існує між будь-якими вершинами, замінними єдиною вершиною в процесі «закорочування» ребер. Але таке припущення може не виконуватися, якщо «закорочувані» ребра мають орієнтацію. Для усунення зазначених недоліків у роботі передбачається алгоритм визначення шляхів з максимальною пропускною здатністю для мереж, описуваних довільними графами, заснований тільки на властивості шляхів графа, що полягає в тім, що пропускна здатність шляху μ_{st} визначається ребром $(i, j) \in \mu_{st}$ з найменшою пропускною здатністю c_{ij}, тобто

$$C(\mu_{st}) = \min_{(i,j \in \mu_{st})} [c_{ij}]. \qquad (5.20)$$

Розв'язання задачі

Нехай граф $G(V,E)$ заданий матрицею пропускних здатностей $C = [c_{ij}]$ і між вершинами *s* і *t* потрібно знайти шлях μ_{st} з максимальною пропускною здатністю $C^*(\mu_{st})$ за умови, що хоча б один шлях між

вершинами s і t існує. Розглянемо алгоритм розв'язання цієї задачі, на кожній ітерації якого будь-яка вершина графа G має мітку $l(v)$, вона може бути постійною або змінною. У першому випадку l(v) – максимальна пропускна здатність шляху μ_{st}, що проходить тільки через вершини з постійними мітками. Таким чином, тимчасова мітка є оцінкою зверху для пропускної здатності шляху μ_{st} і, ставши на деякій ітерації постійною, вона залишається такою до кінця роботи алгоритму. З кожною вершиною v графа $G(V, E)$ будемо зв'язувати ще одну мітку $\Theta(x,v)$. На кожній ітерації x є номером вершини, що передує v у шляху μ^*_{st}, що має максимальну пропускну здатність усіх μ_{st}, що проходять через вершини, які одержали до даного моменту постійні мітки. Після того як вершина t одержить постійну мітку, за допомогою міток $\Theta(x,v)$ легко вказати послідовність вершин, що становлять шлях μ^*_{st}. Перед початком першої ітерації алгоритму вершина s має постійну мітку $l(s)=\infty$, а мітки інших вершин дорівнюють 0, і ці мітки тимчасові. Нехай p – вершина, що одержала постійну мітку $l(p)$ на попередній ітерації. Переглядаємо всі вершини $v \in \tilde{A}(\eth)$, що мають тимчасові мітки, з метою збільшення цих міток. Мітка $l(v)$ заміняється влучною, що дорівнює

$$l(v) = \max \{l(v), \min \{l(p), C(p,v)\}\}. \qquad (5.21)$$

У випадку, якщо у виразі (5.32) максимальним є елемент min $\{l(p), C(p,v)\}$, будемо говорити, що вершина v одержала свою мітку з вершини p і вважаємо $x = p$ в $\Theta(x,v)$. Якщо ж максимальним елементом у виразі (5.21) є $l(v)$, то її мітка Θ не змінюється. Алгоритм закінчує роботу, коли мітка $l(t)$ стає постійною. Сам шлях ідентифікується за допомогою міток Θ (x,v), де x визначає номер вершини, з якої вершина v одержала постійну мітку.

Алгоритм А визначення шляху з максимальною пропускною здатністю

Крок 1. Припустити $l(s) = \infty$ і вважати цю мітку постійною; припустити l(v) = 0 для всіх $v \in V(v \neq s)$ і вважати ці мітки тимчасовими ; p: = s; у мітках $\Theta(x, v)$ x: = 0.

Крок 2. Всім $v \in \Gamma(p)$ з тимчасовими мітками привласнюємо нові значення $l(v)$ = max $\{l(v), \min \{l(p), C(p,v)\}\}$, при цьому, якщо максимальним елементом є min $\{l(p), C(p,v)$, то $x = p$, інакше мітки Θ залишаються незмінними. Інші мітки не змінюються.

Крок 3. Нехай V'' - множина вершин з тимчасовими мітками $l(v)$, знаходимо таку вершину v^*, що $l(v^*) = \max\limits_{v \in V} l(v)$, і вважаємо постійною влучною вершиною v^*.

Крок 4. $p := v^*$, якщо $p = t$, то $l(t) = C^*(\mu^*_{st})$, і шлях μ^*_{st}, обумовлений мітками Θ, є шляхом з максимальною пропускною здатністю $C^*(\mu^*_{st})$, інакше переходимо до виконання кроку 2.

Твердження 5.1. Алгоритм *A* будує в графі *G*(*V, E*) з *n* вершинами шлях з максимальною пропускною здатністю $C^*(\mu^*_{st})$ за час $0(n^2)$.

Це твердження можна довести по індукції.

Нехай вершина *v* одержала свою постійну мітку на *k*-й ітерації, тобто після *k*-го виконання кроку 3. При $k = 1$ справедливість твердження очевидна: вона випливає з того факту, що максимальна пропускна здатність шляху визначається ребром з мінімальною пропускною здатністю (5.20). Припустимо, що воно є правильним для вершин, що одержали свої постійні мітки на ітераціях 2, 3, ... *k* -1. Позначимо через $\bar{\mu}_{st}$ шлях, побудований алгоритмом *A* в результаті *k*–ї і ітерації, а через μ^*_{st} – шлях з максимальною пропускною здатністю $C^*(\mu^*_{st})$ у графі *G*(*V, E*). За визначенням міток, $C(\bar{\mu}_{st}) = l(v)$. Нехай V_1 і V_2 – множини вершин, що мають відповідно постійні й тимчасові мітки перед початком *k*-ї ітерації. Розглянемо дві можливі ситуації:

а) шлях μ^*_{st} містить вершини з V_2, нехай $\bar{v}$ – перша така вершина (рахуючи від *v*), що належить μ^*_{st}, а вершина v' передує $\bar{v}$ у шляху μ_{sv}, тобто $(v', \bar{v}) \in E \mid \mu_{sv}$. Позначимо через $\mu^*_{s\,\bar{v}}$ – частину шляху μ_{sv}. За припущенням про індукцію $l(v') = C^*(\mu_{sv'})$, тому

$$C^*(\mu^*_{sv'}) \geq C(\mu_{s\,\bar{v}}) = l(\bar{v}) \geq C^*(\mu_{sv}). \qquad (5.22)$$

Оскільки $l(v')$ – тимчасова мітка, а постійна $l(v)$ вершини v вибирається на *k*-й ітерації як найбільша з тимчасових, то $l(v) \geq l(v)$, об'єднавши цю нерівність із виразом (5.22), одержимо

$$l(v) = C(\mu_{sv}) \geq C(\mu_{s\,\bar{v}}) \geq C^*(\mu^*_{sv'}) \geq C^*(\mu^*_{sv}),$$

тобто шлях μ_{sv}, побудований алгоритмом *A*, є шляхом з максимальною пропускною здатністю;

б) всі вершини шляху μ^*_{sv} входять у V_1, нехай v' і v'' – такі вершини, що $(v', v) \in E \mid \mu_{sv}$ і $(v'', v) \in E \mid \bar{\mu}_{sv}$, тому, якщо $v' = v''$, відразу одержимо нерівність $l(v) = C(\bar{\mu}_{sv}) \geq l(v') \geq C^*(\mu^*_{sv})$. Припустимо тепер, що $v' \neq v''$. Оскільки *v* одержує мітку $l(v)$ з v'', а не з v', то

$$l(v) = C(\bar{\mu}_{sv}) \geq l(v'') \geq C^*(\mu^*_{sv}).$$

У такий спосіб і у випадку б) шлях, що будується алгоритмом А, є шляхом з максимальною пропускною здатністю.

Оцінимо трудомісткість алгоритму. Обчислювальні витрати максимальні, коли вершина t одержує постійну мітку останньою, а граф G є повним. У цьому випадку кількість ітерацій алгоритму дорівнює (n-1), тобто кожний із кроків (2-4) виконується (n-1) раз. Очевидно, що крок 4 виконується за час $O(1)$, а для виконання кожного із кроків (2,3) досить часу $O(n)$. Побудову шляху за допомогою міток Θ можна здійснити не більш, ніж за $O(n)$ операцій. Таким чином, загальна складність алгоритму побудови шляхи з максимальною пропускною здатністю не перевищує $O(n^2)$.

Запропонований алгоритм, очевидно, має мінімальну тимчасову складність у послідовному варіанті його реалізації, оскільки кожна вершина в ньому розглядається один раз, а щоб розв'язати аналізовану задачу, необхідно переглянути всі вершини графа.

5.10. Паралельна реалізація алгоритму визначення шляхів з максимальною пропускною здатністю

Покажемо тепер, що алгоритм A може бути ефективно розпаралелений на основі рангового підходу до аналізу шляхів у графі $G(V, E)$, що полягає в розробленні процедури A_1, що дозволяє одночасно формувати множини шляхів рангу $r = 1$, $r = 2$, ... і т. д., і розв'язувати при цьому поставлену задачу.

Під рангом шляху μ^r_{st} розуміється кількість ребер, що утворять цей шлях. Розглянемо деякий довільний повний граф $G(V,E)$. Виділимо в ньому дві вершини s і t, множину шляхів $M_{st}=\{\mu^r_{st}\}$, що існує в $G(V,E)$ між цією парою вершин, можна представити у вигляді такого об'єднання підмножин:

$$M_{st} = M_{st}^{r=1} \cup M_{st}^{r=2} \cup \ldots \cup M_{st}^{r=n-1},$$

де M_{st}^r – множина шляхів рангу r між вершинами s і t.

Побудуємо його геометричний еквівалент у такий спосіб. Виділимо вершину s, утворимо ярус із усіх вершин, що залишилися, графа $G(V, E)$ і ще n - 2 таких яруси, причому так, щоб однойменні вершини утворили горизонтальні лінійки. З'єднаємо вершини між ярусами за типом повного двочасткового графа. При цьому одержимо стягнуте дерево всіх шляхів D графа $G(V, E)$.

Стягнуте дерево всіх шляхів D може бути отримане з дерева всіх шляхів з вершини s шляхом стягування однойменних вершин на ярусах, впорядковуючи їх у горизонтальні лінійки. Множини шляхів, задані обома деревами $D(1)$ і D (1), однакові, але для прочитання шляхів в D (1) на кожній горизонтальній лінійці дозволяється бувати тільки один раз. У вершинах i дерева D множини шляхів M_{si}^r. Шлях μ^r_{si} у D може бути

продовжений до шляху рангу $r =: r + 1$ до вершини j, якщо вершина j не входить у шлях μ^r_{si} і в графі G є зв'язок між вершинами i і j, тобто

$$\mu^{r+1}_{sj} = \mu^r_{si} \cup (i,j). \qquad (5.23)$$

Довжину шляху μ^{r+1}_{sj} рангу r+1 позначимо $d^{r+1}[j]$. Оскільки граф D є еквівалентним поданням вихідного графа $G(V, E)$, у якого ребрам відповідають пропускні здатності c_{ij}, то ці самі пропускні здатності відповідають і ребрам графа D. Довжину ребра в D визначимо в такий спосіб: якщо в ребрі (i,j) у вершину і входить потік u, то довжина цього ребра дорівнює $d[i, j] = \min [u, c_{ij}]$, де c_{ij} – пропускна здатність ребра (i, j).

Під потоком будемо розуміти додатні числа $\{u_{ij}\}$, задані на ребрах графа, що характеризують об'єми інформації, які можуть передаватися в мережі.

Розглянемо довільний шлях μ_{st}, припускаючи, що в S надходить потік $u = \infty$ нескінченно великої величини, тоді оскільки вхідний потік у вершину j у шляху μ_{st} визначається пропускною здатністю c_{ij} попереднього ребра в цьому шляху, то довжина шляху $d(\mu_{st}) = d[t] = \min\limits_{c_{ij} \in \mu_{st}} \{c_{ij}\}$, тобто довжина шляху в μ_{st} дорівнює максимальній пропускній здатності шляху μ_{st}. Якщо до вершини i існують шляхи $\mu^{r=1}_{si}, \mu^{r=2}_{si}, ..., \mu^{r=n-1}_{si}$, то шлях з максимальною довжиною

$$d[\mu^{*r}_{si}] = \max_r \{d(\mu^{r=1}_{si}), d(\mu^{r=2}_{si}), ..., d(\mu^{r=n-1}_{si})\}$$

є шляхом з максимальною пропускною здатністю між вершинами s та i. Таким чином, задачу визначення шляхів з максимальною пропускною здатністю в графі G(V, E) можна розглядати як задачу визначення найдовших шляхів у D між вершиною s і всіма іншими вершинами графа D з урахуванням уведеного поняття довжини шляхи. Для знаходження найдовших шляхів у D використовуємо процедуру A1.

Процедура A_1

Крок 1. З вершини s будуємо множини шляхів рангу r = 1 і визначаємо їхні довжини $d^{r=1}(i)\, i = \overline{(1, n-1)}$, припускаючи, що у вершину s утікає потік u = ∞. Виділяємо серед $\{d^{r=1}(i)\}$ шлях $\mu^{*r=1}_{si}$ з максимальною довжиною $d^{r=1}_{\max}[\mu^{r=1}_{si}]$ і вершину i виключаємо з подальшого аналізу в D.

Крок 2. Формуємо на основі шляху μ^{*r}_{si} попереднього кроку шляхи $\mu^{r=r+1}_{sj} = \mu^{*r}_{si} \cup (i, j)$, визначаємо їхні довжини, порівнюємо їх з довжинами шляхів попереднього рангу й виділяємо шляхи максимальної довжини на основі такого рекурентного співвідношення:

$$d^{r,r+1}_{\max}[j] = \max \{d^r[j], \min(d^r[i], c_{ij})\}.$$

Крок 3. Виділяємо в підмножині $\{\mu_{si}^{r:=r+1}\}$ шлях максимальної довжини й виключаємо вершину i з подальшого аналізу.

Крок 4. Перевіряємо, чи всі вершини виключені з розгляду в D, якщо ні, то переходимо до кроку 2. Інакше процедура A_1 закінчує роботу.

Неважко побачити, що процедури A і A_1 еквівалентні, оскільки рекурентне співвідношення на кроці 2 точно відповідає співвідношенню (5.17) при розміщенні міток процедурою A. Єдиною відмінністю тут є те, що процедура A_1 зорієнтована на побудову дерева шляхів з максимальною пропускною здатністю між вершиною s та іншими вершинами графа $G(V, E)$. Цю різницю легко усунути, якщо модифікувати крок 4 алгоритму A_1 так, щоб алгоритм закінчував роботу після одержання всіма вершинами постійних міток. Перевагою процедури A_1 є те, що вона дозволяє формувати шляхи наступного рангу $\mu_{sj}^{r:=r+1} = \mu_{si}^{*r} \cup (i, j)$ одночасно для всіх $i = \overline{(1, n-1)}$ на $(n-1)$ процесорі. У кожній вершині D буде сформовано не більше двох шляхів, тобто одночасно можна буде провести не більше $(n-1)$ операцій порівняння й далі вибрати максимальний елемент у масиві, що складається з n – 1 числа. Отже, для виконання кроків 1–2 на $n-1$ процесорі буде потрібно $O(1)$ часу, а крок 3 зажадає $O(\log_2(n-1))$ часу. Оскільки кроки 1-3 доведеться повторити $(n-1)$ раз, то загальна складність алгоритму не перевищить $O((n-1)\log_2(n-1)) = O(n\log_2 n)$. Якщо ж розпаралелити і операцію виділення максимального елемента в масиві на кроці 3, застосовуючи програмувальні логічні матриці, то загальна складність знизиться до $O(n)$. Приклад розв'язання задачі побудови дерева шляхів з максимальною пропускною здатністю від вершини 1 графа G (рис. 5.17) на основі процедури A_1 наведені в табл. 5.5, у якій показано етапи формування й виділення шляхів процедурою A_1. Нехай граф G мережі має вигляд, як на рис. 5.17, у кружках проставлені пропускні здатності ребер мережі й потрібно побудувати дерево шляхів з максимальними пропускними здатностями до всіх інших вершин графа.

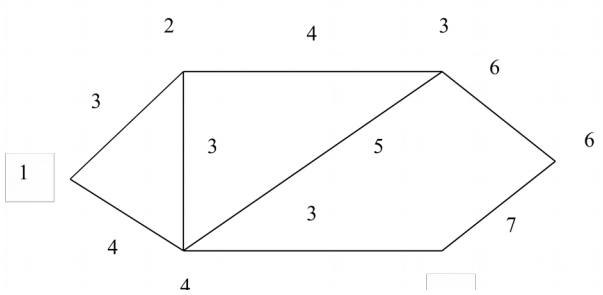


Рис. 5.17. Граф G

Тоді процес його побудови можна представити у вигляді табл. 5.5.

Таблиця 5.5

Процес визначення шляхів з максимальною пропускною здатністю

12(3) 2	12(3) 142(3) 2	12(3) 1432(4) 2	- 2	- 2
– 3	143(4)∗ 3	– 3	– 3	– 3
14(4)∗ 4	– 4	– 4	– 4	– 4
5	145(3) 5	145(3) 5	145(3) 5	145(3) 14365(4) 5
– 6	– 6	1436(4) 6	1436(4)∗ 6	– 6

Якщо порівняти процедуру A_1 з паралельною реалізацією алгоритму Дейкстри, запропонованою в даному розділі, то неважко побачити, що вони можуть бути реалізовані на одній і тій самій архітектурі ~~ПВС~~ ПОС циклічного типу. По шляхах, позначених зірочкою, будуємо дерево шляхів з максимальною пропускною здатністю (рис. 5.18).

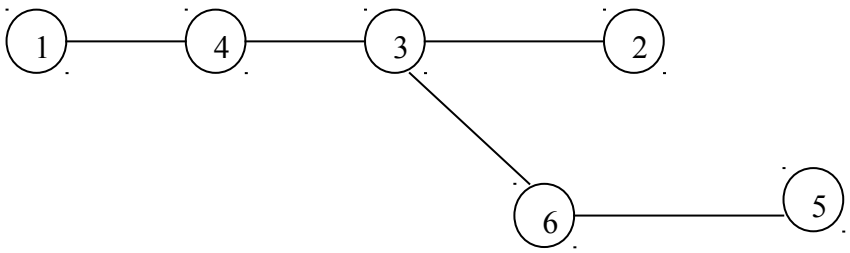


Рис. 5.18. Дерево шляхів з вершини 1

5.11. Оцінка сумарної пропускної здатності шляхів телекомунікаційної мережі на основі рангового підходу

У роботах [16, 17, 22] оцінка пропускної здатності шляхів у мережах базується на алгоритмах для двополюсних мереж, заснованих на теоремі «про максимальний потік і мінімальний розріз». Більшість реальних мереж не є двополюсними мережами, тому можливості використання даних методів оцінки сумарної пропускної здатності шляхів мереж досить обмежені.

Для визначення сумарної пропускної здатності багатополюсних мереж, тобто мереж, у яких потоки передаються одночасно між декількома парами вузлів мережі, у роботах [3, 4] була зроблена спроба поширити умови можливості передачі потоку у двополюсній мережі на багатополюсні мережі з постійною системою, що змінюється в часі вимог на передачу потоків. У цих роботах стверджується, що для здійснення одночасної

передачі декількох потоків, заданих системою вимог Ф = $\{\varphi_{ij}\}$, у багатополюсній мережі необхідно й досить, щоб величина будь-якого перетину мережі була більше або дорівнювала сумі вимог на передачу потоків між вузлами, поділюваними цим перетином. Для будь-якого перетину S_γ мережі має бути справедливою нерівність

$$\sum_{\substack{k\in\gamma \\ l\notin\gamma}} \varphi_{kl} \le V_\gamma , \qquad (5.24)$$

де V_γ – величина перетину S_γ, що визначається сумою пропускних здатностей ребер, що утворять цей перетин,

$$V_\gamma = \sum_{\substack{i\in\gamma \\ j\in(H-\gamma)}} c_{ij} ,$$

де c_{ij} – пропускна здатність ребра β_{ij};

$H = \{h\}$ ($h = 1, 2, \ldots, n$) – множина вузлів мережі.

Дослідження, проведені в роботах [11, 16, 17], показали, що умова (5.24) є достатньою тільки для мереж, у яких жоден шлях, використовуваний для передачі цих потоків, не перетинає мережі більше одного разу [156]. Ця обставина трохи обмежує використання згаданих вище робіт для оцінки сумарної пропускної здатності шляхів у багатополюсних мережах. Крім того, при обмежених комутаційних можливостях устаткування вузлів мережі, коли для передачі потоків може бути використана тільки частина шляхів між вузлами мережі, визначення величини V_γ, як відзначається у роботі [4], стає досить трудомістким процесом, який необхідно здійснювати в масштабі реального часу, обумовленого циклом відновлення інформації в мережі.

Розглянемо можливість розв'язання цієї задачі за допомогою пропонованого рангового підходу на основі методу, запропонованого в роботах [143, 146]. Для визначення сумарної пропускної здатності шляхів у ній будемо виділяти сукупність шляхів $M = \{\mu_1, \mu_2, \ldots, \mu_r\}$ між парами вузлів мережі. (Для виділення шляхів можна скористатися одним з методів визначення найкоротших шляхів на основі рангового підходу, наведених у розд. 3 даного посібника). За заданою сукупністю шляхів мережі будується матриця шляхів $P = |p_{ij}|$, кожний рядок матриці Р відповідає одному зі шляхів μ_i мережі, а кожний стовпець – одному з ребер β_i мережі. (Для зручності запису ребра мережі в цьому випадку перенумеровані і подвійний індекс ребра β_{ij} замінено одним індексом β_l). Елемент матриці P, що стоїть на перетині довільного l-го стовпця й к-го рядка, $p_{kl} = 1$, якщо ребро β_l належить шляху μ_k, і $p_{kl} = 0$ в інших випадках. Зазначена вище матриця шляхів $P = |p_{ij}|$ і матриця пропускних здатностей ребер мережі

$C = |c_{ij}|$ служать вихідними даними для визначення сумарної пропускної здатності шляхів мережі. При визначенні сумарної пропускної здатності мережі використовується поняття «покриття» сукупності шляхів. Покриттям сукупності шляхів $M = \{\mu_1, \mu_2, ..., \mu_r\}$ називається така множина ребер $B = \{\beta_1, ..., \beta_k\}$, у якій в кожному зі шляхів сукупності M міститься хоча б одне ребро із множини B. (Іншими словами, якщо з мережі видалити ребра, що входять у множину B, то всі шляхи сукупності M виявляться обірваними). Величиною покриття Q називається сума пропускних здатностей ребер, що входять у це покриття. Описуваний спосіб визначення сумарної пропускної здатності шляхів мережі заснований на теоремі, наведеній у роботі [4], у якій стверджується, що «максимальна сумарна пропускна здатність заданої сукупності шляхів $M = \{\mu_1, \mu_2, ..., \mu_r\}$ розглянутої мережі не може бути більше величини мінімального покриття цих шляхів», тобто

$$\sum_{k=1}^{r} f_{\mu k} \le Q_{\min},$$

де $f_{\mu k}$ – пропускна здатність шляху $\mu_k \in M$.

У такий спосіб задача визначення сумарної пропускної здатності шляхів мережі зводиться до визначення мінімального покриття за матрицею P. Величину мінімального покриття Q_{min} можна визначити безпосередньо за матрицею шляхів $P = |p_{ij}|$ на основі алгоритмів розв'язання задачі про мінімальне покриття, наведених у розд. 2 даного посібника, які дозволяють розв'язувати такі задачі досить великої розмірності в масштабі реального часу.

Вправи

1. Покажіть, що на паралельних структурах циклічного типу з n процесорами можна розв'язувати одну задачу за час, що дорівнює її розв'язанню на n^2 процесорах у матричних структурах.

2. Побудуйте структуру циклічного типу для множення двох матриць розмірності 3x3 виконуючи множення за дев'ять кроків і за один крок.

3. Порівняйте тимчасову складність паралельних реалізацій алгоритмів Форда-Фалкерсона й Дейкстри.

4. Порівняйте за тимчасовою складністю алгоритми Френка й Фриша для визначення шляхів з максимальною пропускною здатністю з ранговим паралельним алгоритмом.

5. Покажіть, як на основі паралельної реалізації алгоритму Форда-Фалкерсона можна побудувати алгоритм знаходження k найкоротших шляхів.

6. Поясніть, що таке ширина паралельної форми алгоритму й що таке

висота паралельної форми алгоритму.

7. Поясніть, що означає вираз «структура алгоритму й структура обчислювальної системи взаємозалежні».

БІБЛІОГРАФІЧНИЙ СПИСОК

1. Вычислительные сети и сетевые протоколы [Текст]: пер. с англ. / Д. Девис, Д. Барбер, У. Прайс, С. Соломонидес. – М.: Мир, 1982. – 562 с.

2. Янбих, Г.Ф. Оптимизация информационно-вычислительных сетей [Текст]: учеб. пособие / Г.Ф. Янбих, Б.А. Столяров. – М.: Радио и связь, 1987. – 232 с.

3. Microsoft Corporation. Компьютерные сети. Учебный курс [Текст]: пер. с англ. – М.: Русская редакция ТОО «Channel Trading Ltd.», 1997. – 696 с.

4. Berezhnaja, M.A. Two-Frames Cubical Calculus for Test Generation [Текст] / M.A. Berezhnaja, G.F. Krivoulja, V.I. Hahanov // Test Automation. – Kharkiv, 1997. – P. 118.

5. Листровой, С.В. Структурные модели вычислительных сетей ИРС и их характеристики [Текст]: учеб. пособие / С.В. Листровой. – Харьков: ХВВКИУ РВ. МО СССР, 1985. – 85 с.

6. Листровой, С.В. Технические средства [Текст]: учеб. пособие / С.В. Листровой, С.А. Атамасов. – Харьков: ХВВКИУ РВ. МО СССР, 1992. – 189 с.

7. Листровой, С.В. Технические средства информационно-расчетных систем [Текст]: учеб. пособие / С.В. Листровой, С.А. Ильин. – Харьков: ХВВКИУ РВ. МО СССР, 1981. – 67 с.

8. Листровой, С.В. Вычислительные сети [Текст] / С.В. Листровой. – Харьков: ХВВКИУ РВ. МО СССР, 1981. – 78 с.

9. Листровой, С.В. Структуры вычислительных сетей и управление потоками информации в них [Текст]: учеб. пособие / С.В. Листровой. – Харьков: ХВВКИУ РВ. МО СССР, 1980. – 67 с.

10. Бережная, М.А. Диагностирование одиночных и кратных неисправностей [Текст] / М.А. Бережная, В.И. Хаханов // АСУ и приборы автоматики. – Харьков: ХТУРЭ, 1997. – Вып. 104. – С. 17-28.

11. Бережная, М.А. Двухтактное кубическое исчисление. Анализ моделей цифровых устройств [Текст] / М.А. Бережная, В.И. Хаханов // АСУ и приборы автоматики. – Харьков: ХТУРЭ, 1997. – Вып. 106. – С. 103-116.

12. Зиновьев, Е.В. Принципы построения системы управления информационными процессами и ресурсами в сети ЭВМ [Текст] / Е.В. Зиновьев // Автоматика и вычислительная техника. –1985. – №3. – С. 45–52.

13. Зиновьев, Е.В. Управление сетевыми информационными процессами и ресурсами [Текст] / Е.В. Зиновьев. – Рига: Зинатне, 1987. –

303 с.

14. Зиновьев, Е.В. Методы управления сетевыми информационными системами [Текст] / Е.В. Зиновьев, А.А. Стрекалев. – Рига: Зинатне, 1991. – 308 с.

15. Зиновьев, Е.В. Управление информационными процессами и ресурсами в вычислительных системах и сетях с учетом тупиковых ситуаций [Текст]: учеб. пособие / Е.В. Зиновьев, А.А. Стрекалев. – Рига: Зинатне, 1983. – 49 с.

16. Зиновьев, Е.В. Конфликтные ситуации в информационных системах [Текст]: учеб. пособие / Е.В. Зиновьев, А.А. Стрекалев. – Рига: Зинатне, 1985. – 166 с.

17. Спенсер, Р. Архитектура связи в распределенных системах [Текст]: пер. с англ. / Р. Спенсер. –М.: Мир, 1981. – 744 с.

18. Якубайтис, Е.А. Информационно-вычислительные сети [Текст]: учеб. пособие / Е.А. Якубайтис. – М.: Финансы и статистика, 1984. – 232 с.

19. Якубайтис, Е.А. Архитектура вычислительных сетей. [Текст]: учеб. пособие / Е.А. Якубайтис. – М.: Статистика, 1980. – 280 с.

20. Бережная, М.А. Кодирование состояний автоматов, реализуемых в матричных структурах [Текст] / М.А. Бережная // АСУ и приборы автоматики. – Харьков: ХТУРЭ, 2001. – Вып. 114. – С. 43–48.

21. Построение сетей ЭВМ [Текст]: пер. с япон. / М. Като, Д. Иимура, М. Токоро, Е. Тома. – М.: Мир, 1988. – 307 с.

22. Дьяченко, В.Ф. Управление на сетях связи [Текст] / В.Ф. Дьяченко, В.Г. Лазарев, Г.Г. Саввин. – М.: Энергия, 1984. – 277 с.

23. Турута, Е.Н. Об одном методе повышения живучести локальных сетей ЭВМ [Текст] / Е.Н. Турута, В.Ш. Ковалев // Автоматика и вычислительная техника. – 1983. – № 5. – С. 42–44.

24. Якубайтис, Е.А. Открытые информационные системы [Текст]: учеб. пособие / Е.А. Якубайтис. – М.: Радио и связь, 1991. – 208 с.

25. Бережная, М.А. Алгоритм синтеза последовательностных схем, реализуемых в матричных структурах [Текст] / М.А. Бережная // АСУ и приборы автоматики. – Харьков: ХТУРЭ, 2001. – Вып. 116. – С. 93–96.

26. Бережная, М.А. Инструментарий САПР микропроцессорных контроллеров [Текст] / М.А. Бережная [и др.] // Материалы VIII-й Междунар. школы-семинара (Алушта, 7-18 сентября 1995 г.). – Алушта, 1995. – С. 126.

27. Самойленко, С.И. Сети ЭВМ [Текст] / С.И. Самойленко. – М.: Наука, 1986. – 180 с.

28. Bochman, G. Formal methods in communication Protocol design [Текст] / G. Bochman, C. Sunshine // IEEE Trans. Commins. – 1980. – Vol. COM-28. – P. 624–631.

29. Бережная, М.А. О системном проектировании RISC-процессоров [Текст] / М.А. Бережная [и др.] // Теория и техника передачи, приема и

обработки информации: тезисы докладов II-й Междунар. конф. (Туапсе, 17-19 сентября 1996 г.). – Харьков, 1996. – С. 239.

30. Бережная, М.А. Исследование базовых архитектур RISC-процессоров [Текст] / М.А. Бережная [и др.] // Теория и техника передачи, приема и обработки информации: тезисы докладов Междунар. конф. (Севастополь, 7-10 октября 1996 г.). – Харьков, 1996. – С. 29.

31. Miroschnik, M.A. Model of influences of sensor reflections on the accuracy of microwave reflectometer [Текст] / M.A. Miroschnik, I. Klyuchnyk, R. Tsekhmistro, Z. Warsza, O. Zaichenko // PAK (Pomiary Automatyka Kontrola). – 2014. – Vol. 60. – P. 223-225.

32. Турута, Е.Н. Обеспечение отказоустойчивости управляющих многомикропроцессорных систем путем перераспределения задач отказавших модулей [Текст] / Е.Н. Турута // Системы управления информационных сетей. – М.: Наука, 1983. – С. 187–198.

33. Назаров, С.В. Операционные системы специализированных вычислительных комплексов: Теория построения и системного проектирования [Текст] / С.В. Назаров. – М.: Машиностроение, 1989. – 400 с.

34. Поляков, Г.А. Автоматизация проектирования сложных цифровых систем коммутации и управления [Текст] / Г.А. Поляков, Ю.Д. Умрихин. – М.: Радио и связь, 1988. – 304 с.

35. Бережная, М.А. К задаче выбора базовой структуры микропроцессоров [Текст] / М.А. Бережная [и др.] // Электроника и молодежь в 21 веке: тезисы докладов I-го Междунар. форума (Харьков, 22-24 апреля 1997 г.). – Харьков, 1997. – С. 229.

36. Блек, Ю. Сети ЭВМ: Протоколы, стандарты, интерфейсы [Текст]: пер. с англ. / Ю. Блек. – М.: Мир, 1990. – 506 с.

37. Киселев, В.Д. Оптимизация восстановительного резервирования информации в сетях ЭВМ [Текст] / В.Д. Киселев, О.В. Ешков // Электронное моделирование. – 1995. – Т. 17, № 3. – С. 53–58.

38. Бондарев, В.Н. Искусственный интеллект [Текст] / В.Н. Бондарев, Ф.Г. Аде. – Севастополь: СевНТУ, 2002. – 615 с.

39. Третьяк, В.Ф. Реализация приближенного алгоритма целочисленного программирования на систолических однородных средах [Текст] / В.Ф. Третьяк, С.В. Кавун, А.Ю. Гуль // Обработка информации и обеспечение надежности систем управления. – Харьков: НАНУ: ПАНИ: ХВУ, 1996. – С. 144–147.

40. Воеводин, В.В. Математические модели и методы в параллельных процесах [Текст] / В.В. Воеводин. – М.: Наука, 1986. – 296 с.

41. Вычислительная техника социалистических стран [Текст] // Компьютер и задачи выбора: сб. статей. – М.: Наука, 1989. – С. 208.

42. Сергиенко, И.В. О формальных методах в компьютерных технологиях [Текст] / И.В. Сергиенко, И.Н. Парасюк, А.И. Проватор //

Кибернетика и системный анализ. – 1998. – № 4.–С. 59–70.

43. Бережная, М.А. О системном подходе к проектированию специализированных систем обработки данных [Текст] / М.А. Бережная [и др.] // Теория и техника передачи, приема и обработки информации: тезисы докладов Междунар. конф. (Туапсе, 18-21 сентября 1995 г.). – Туапсе, 1995. – С. 157.

44. Воеводин, В.В. Математические проблемы освоения суперЭВМ [Текст] / В.В. Воеводин // Вычислительные процессы и системы. – М.: Наука, 1985. – Вып. 2. – С. 73.

45. Глушков, В.М. О некоторых проблемах решения задач на ЭВМ с параллельной организацией вычислений [Текст] / В.М. Глушков, И.Н. Молчанов // Кибернетика. – 1981. – № 4. – С. 82-87.

46. Гун, С. Систолические и волновые процессоры для высокопроизводительных вычислений [Текст] / С. Гун // Труды ИИЕР. – 1984. – № 7. – С. 133.

47. Каляев, А.В. Многопроцессорные суперсистемы с программируемой архитектурой, основанные на принципе потока данных [Текст] / А.В. Каляев // Вычислительные процессы и системы. – М.: Наука, 1985. – Вып. 2. – С. 140.

48. Котов, В.Е. О связи алгоритмических и архитектурных аспектов параллельных вычислений [Текст] / В.Е. Котов // Вычислительные процессы и системы. – М.: Наука, 1983. – Вып. 1. – С. 54.

49. Марчук, Г.И. Модульная асинхронная развиваемая система. Ч. 1. Предпосылки направления развития архитектуры вычислительных систем / Г.И. Марчук, В.Е. Котов. – М., 1980. – 27 с. – (Препринт / ВЦ СО АН СССР; № 86).

50. Бережная, М.А. Диагностирование ПЛИС на основе моделей клеточных автоматов [Текст] / М.А. Бережная [и др.] // Перспективные системы управления на железнодорожном, промышленном и городском транспорте: тезисы докладов XV–й Междунар. школы-семинара (Алушта, 13-23 сентября 2002 г.) – Харьков, 2002. – С. 33.

51. Додонов, А.Г. Организация специализированных вычислительных структур для решения комбинаторно-логических задач на сетях с применением микроЭВМ [Текст] / А.Г. Додонов // Электронное моделирование. – 1985. – Т. 7, № 6. – С. 14–20.

52. Бережная, М.А. Отказоустойчивые системы управления на основе микроконтроллеров [Текст] / М.А. Бережная, Л.В. Дербунович, В.С. Суздаль, И.Н. Темников // Проблемы автоматизированного электропривода: Вестник НТУ “ХПИ”. Серия “Электротехника, электроника и электропривод”. – Харьков, 2003. – Вып. 12. – Т. 1. – С. 218–220.

53. Бережная, М.А. Синтез синдромно-тестируемых программных логических контроллеров. Ч. 1. Метод тестирования [Текст] / М.А. Бережная, Д.А. Татаренко // Інформаційно – керуючі системи на

залізничному транспорті. – Харьков, 2004. – №3 (47). – С. 30–37.

54. Самофалов, К.Г. Пространственно-временная модель планирования работ в параллельной вычислительной системе [Текст] / К.Г. Самофалов, В.П. Симоненко // Электронное моделирование. – 1996. – Т. 18, № 4. – С. 22–28.

55. Богатырев, В.А. Протоколы динамического перераспределения запросов в распределенных вычислительных системах [Текст] / В.А. Богатырев // Электронное моделирование. – 1996. – Т. 18, № 3. – С. 24–27.

56. Баранов, В.И. Экстремальные комбинаторные задачи и их приложения [Текст] / В.И. Баранов, Б.С. Стечкин. – М.: Наука, 1989. – 160 с.

57. Белколин, В.Е. Анализ и синтез систем автоматизированного управления на ЭВМ. Алгоритмы и программы [Текст] / В.Е. Белколин, П.И. Чинаев. – М., 1989. – 235 с.

58. Вышенчук, И.М. Алгоритмические операционные устройства и суперЭВМ [Текст] / И.М. Вышенчук, Н.В. Черкасский. – К.: Техника, 1990. – 197 с.

59. Konovalov, A. Virtual Shared Files: Towards User-Friendly Inter-Process Communications [Текст] / Alexandr Konovalov, Victor Samofalov, Sergei Scharf // Parallel computing technologies: 5th international conference; PaCT-99, St. Petersburg, Russia, September 6-10, 1999, Victor Malyshkin (ed.), Lecture Notes in Computer Science. – Vol. 1662. – P. 223–228.

60. Симоненко, В.П. Обоснование выбора метода динамической диспетчеризации работ в распределенных вычислительных системах [Текст] / В.П. Симоненко // Электронное моделирование. – 1998. – Т. 20, № 6. – С. 14–32.

61. Симоненко, В.П. Модель распределения задач в неоднородных системах распределенной обработки данных [Текст] / В.П. Симоненко // Электронное моделирование. – 1998. – Т. 20, № 7. – С. 45–56.

62. Замбицкий, Д.К. Алгоритмы решения оптимизационных задач на сетях [Текст] / Д.К. Замбицкий, Д.Д. Лозовану. – Кишинев: «Штиница», 1983. – 116 с.

63. Надежность и эффективность в технике [Текст] / под ред. В.Ф. Уткина. – М.: Машиностроение, 1988. – Т. 3. – 328 с.

64. ГОСТ 27.002-89. Надежность в технике. Термины и определения [Текст]. – М.: Издательство стандартов, 1989. – 25 с.

65. ГОСТ 24.701-86. Надежность автоматизированных систем управления. Основные положения [Текст]. – М.: Издательство стандартов, 1986. – 78 с.

66. ГОСТ 27.004-85. Системы технологические. Термины и определения [Текст]. –М.: Издательство стандартов, 1985. – 68 с.

67. ГОСТ 24.702-85. Эффективность автоматизированных систем управления. Основные положения [Текст]. – М.: Издательство стандартов,

1985. – 35 с.

68. ГОСТ 24.703-85. Типовые проектные решения в АСУ. Основные положения [Текст]. – М.: Издательство стандартов, 1985. – 47 с.

69. Харченко, В.С. Живучесть и безопасность систем управления летательных аппаратов и комплексов. Ч. 1. Основные понятия и модели [Текст]: учеб. пособие / В.С. Харченко, П.Е. Марков. – М.: МОУ, 1994. – 68 с.

70. ГОСТ 27.002-89. Надежность в технике. Термины и определения [Текст]. – М.: Издательство стандартов, 1989. – 25 с.

71. Бережная, М.А. Диагностические модели многопроцессорных систем управления [Текст] / М.А. Бережная, Л.В. Дербунович, В.С. Суздаль, Д.А. Татаренко // Інформаційно-керуючі системи на залізничному транспорті. – Харків, 2004. – №6 (50). – С. 33–37.

72. Бережная, М.А. Синтез синдромно-тестируемых логических контроллеров. Ч. 2. Примеры синтеза легкотестируемых схем [Текст] / М.А. Бережная, Д.А. Татаренко // Інформаційно – керуючі системи на залізничному транспорті. – Харьков, 2005. – №3 (53). – С. 49–53.

73. Бережная, М.А. Синтез легкотестируемых схем на основе полинома Рида-Майллера [Текст] / М.А. Бережная [и др.] // Інформаційні технології: наука, техніка, технологія, освіта, здоров'я: XIII-та міжнар. наук.-практ. конф. (Харків, 19-20 травня 2005 р.). – Харків, 2005. – С. 34.

74. Бережная, М.А. Диагностирование микроэлектронных устройств на основе сигнатурно-синдромного сжатия данных [Текст] / М.А. Бережная [и др.] // Прикладная радиоэлектроника. Состояние и перспективы развития: II-й Междунар. радиоэлектронный форум МРФ–2005 (Харьков, 19-23 сентября 2005 г.). – Харьков: ХНУРЭ, 2005. – С. 115-118.

75. Зайченко, Ю.П. Исследование операций [Текст] / Ю.П. Зайченко. – К.: Вища школа, 1988. – 552 с.

76. Калинин, В.Н. Теория систем и оптимального управления. Ч. 2. Понятия, модели и алгоритмы оптимального выбора [Текст] / В.Н. Калинин, Б.А. Резников, Е.И. Варакин. – М.: МО СССР, 1987. – 590 с.

77. Бережная, М.А Синдромно-сигнатурное тестирование микроэлектронных устройств [Текст] / М.А. Бережная, М.Г. Рыжикова // Физические и компьютерные технологии: XI-я Междунар. науч.-техн. конф. (Харьков, 2-3 июня 2005 г.). – Харьков: ХНУРЭ, 2005. – С. 113.

78. Пападимитриу, Х. Комбинаторная оптимизация. Алгоритмы и сложность [Текст]: пер. с англ. К. Стайглиц. – М.: Мир, 1985. – 512 с.

79. Рейнгольд, Е. Комбинаторные алгоритмы. Теория и практика [Текст] / Е. Рейнгольд, Ю. Нивергельт, Н. Део. – М.: Мир, 1980. – 476 с.

80. Бережная, М.А. Синтез комбинационных схем в базисе полиномиальных форм [Текст] / М.А. Бережная, М.Г. Рыжикова, Д.А. Татаренко // Радиоэлектроника и информатика. – 2005. – № 3 (32).– С. 103–109.

81. Berezhna, M. Pseudoexhaustive tpg based on nonlinear feedback shift

registers [Текст] / M. Berezhna, L. Derbunovsch, M. Ryzhskova, D. Tatarenko // Інформаційно – керуючі системи на залізничному транспорті. – Харків, 2005. – №5. – С. 54-59.

82. Гуляницкий, Л.Ф. О методах дискретной оптимизации для многопроцессорных вычислительных комплексов [Текст] / Л.Ф. Гуляницкий, И.В. Сергиенко, А.Н. Ходзинский // Кибернетика. – 1988. – № 4. – С. 26–33.

83. Емеличев, В.А. Метод построения последовательности планов для решения задач дискретной оптимизации [Текст] / В.А. Емеличев, В.И. Комлик. – М.: Наука, 1981. – 208 с.

84. Бережная, М.А. Генераторы тестов для встроенного самотестирования дискретных устройств [Текст] / М.А. Бережная, Л.В. Дербунович, М.Г. Рыжикова, Д.А. Татаренко // Проблемы автоматизированного электропривода: Вестник НТУ "ХПИ". – Харьков, 2005. – Вып. 15. – С. 218–221.

85. Бережная, М.А. Диагностирование цифровых систем в свете современных электронных технологий [Текст] / М.А. Бережная, Л.В. Дербунович, В.С. Суздаль // Информатика и моделирование: Вестник НТУ "ХПИ": сб. науч. трудов: тематический выпуск. – Харьков: НТУ "ХПИ", 2005. – № 56. – С. 3–8.

86. Бережная, М.А. Декомпозиционный метод синтеза легкотестируемых дискретных устройств [Текст] / М.А. Бережная, М.Г. Рыжикова, С.Г. Карпенко // Радиоэлектроника и молодежь в XXI веке: 10-й Юбилейный междунар. молодежный форум (Харьков, ХНУРЭ, 10-12 апреля 2006 г.). – Харьков: ХНУРЭ, 2006. – С. 149.

87. Бережная, М.А. Синтез тестопригодных схем путем устранения функционально-структурной избыточности [Текст] / М.А. Бережная, А.А. Андрусевич, Я.Ю. Королева [и др.] // Вісті Академії наук України. Науково-технічний та громадянський часопис Президії Академії інженерних наук України. Машинобудування та прогресивні технології. – Харків: ХНАУ, 2006. – №3 (30).– С. 188–192.

88. Михалевич, В.С. Методы последовательной оптимизации в дискретных сетевых задачах оптимального распределения ресурсов [Текст] / В.С. Михалевич, А.И. Кукса. –М.: Наука, 1983. – 208 с.

89. Михалевич, В.С. Исследование методов решения оптимизационных задач и их приложения [Текст] / В.С. Михалевич, И.В. Сергиенко, Н.З. Шор // Кибернетика. – 1981. – № 4. – С. 89–113.

90. Олешко, М.В Алгоритм для решения задач целочисленного программирования с булевыми переменными, использующий вероятностные оценки [Текст] / М.В. Олешко, В.П. Шило // Программное обеспечение экстремальных задач и пакеты прикладных программ. – К.: Ин-т кибернетики АН УССР, 1982. – С. 110–113.

91. Сергиенко, И.В. Математические модели и методы решения задач дискретной оптимизации [Текст] / И.В. Сергиенко. – К.: Наук. думка, 1988.

– 472 с.

92. Сергиенко, И.В. Модели и методы решения на ЭВМ комбинаторных задач оптимизации [Текст] / И.В. Сергиенко, М.Ф. Каспшицкая. – К.: Наук. думка, 1981. – 288 с.

93. Сергиенко, И.В. Приближенные методы решения дискретных задач оптимизации [Текст] / И.В. Сергиенко, Т.Т. Лебедева, В.А. Рощин. – К.: Наук. думка, 1980. – 276 с.

94. Сергиенко, И.В. Полиномиальный асимптотический ε-оптимальный алгоритм случайного поиска для задач булевого линейного программирования [Текст] / И.В. Сергиенко, В.П. Шило //Докл. АН УССР. Серия А. – 1987. – № 3. – С. 70–72.

95. Бережная, М.А. Декомбинационный метод синтеза управляющих устройств [Текст] / М.А. Бережная [и др.] / Інформаційні технології: наука, техніка, технологія, освіта, здоров'я. Секція 8. Мікропроцесорна техніка в автоматиці та приладобудуванні: 14 міжнар. наук.-практ. конференція. (Харків, 18-19 травня 2006 р.). – Харків, 2006. – С. 102.

96. Berezhna, M.A. Fault tolerant multiprocessors control system for manufacturing process of large crystals growth [Текст] / M.A. Berezhna, L.V. Derbunovsch, V.S. Suzdal // Інформаційно-керуючі системи на залізничному транспорті. Перспективні компютерні управляющі и телекомунікаційні системи: матеріали 20-ї міжнар. конф. (Алушта, 19-21 вересня 2006 р.). – Харків, 2006. – №4 (додаток). – С. 19-20.

97. Бережная, М.А. Синтез дискретных устройств методом последовательной декомпозиции автоматных моделей [Текст] / М.А. Бережная, Л.В. Дербунович, Я.Ю. Королева, М.Г. Рыжикова // Вестник НТУ "ХПИ": сб. науч. трудов. Тематический выпуск: Информатика и моделирование. – Харьков: НТУ "ХПИ", 2007. – № 36. – С. 16–25.

98. Сергиенко, И.В. Рестарт-технология решения задач дискретной оптимизации [Текст] / И.В. Сергиенко, В.П. Шило, В.А. Рощин // Кибернетика и системный анализ. – 2000. – № 5. – С. 32–40.

99. Королев, А.В. Эффективность параллельных алгоритмов оптимизации вычислительного процесса [Текст] / А.В. Королев, С.В. Листровой, В.Ф. Третьяк // Информационно-управляющие системы на железнодорожном транспорте. – 1997. – № 1. – С. 85–91.

100. Листровой, С.В. О возможности решения задач оптимального распределения ресурсов при управлении сложными системами в реальном масштабе времени [Текст] / С.В. Листровой, В.Н. Хрин // Техническая кибернетика: Изв. АН России. – 1992. – № 4. – С. 125–133.

101. Листровой, С.В. Параллельные алгоритмы оптимизации вычислительного процесса для задач булевого программирования [Текст] / С.В. Листровой, В.Ф. Третьяк, Е.С. Листровая // Электронное моделирование. – 1998. – № 5.– С. 23–33.

102. Голубничий, Д.Ю. Математические модели и методы в вычислительных системах и сетях ЭВМ [Текст] / Д.Ю. Голубничий,

С.В. Листровой, В.Ф. Третьяк, А.Ю. Гуль; под ред. С.В. Листрового. – Харьков: ХВУ, 1998. – 191 с.

103. Koroljov, A.V. An optimal planning of algorithms in computer system [Текст] / A.V. Koroljov, O.M. Krjuchov, S.V. Listrovoy, V.F. Tretjak // Theses of conference “Mathematical modeling and information technologies”. – Belgorod, 1997. – P. 32–33.

104. Listrovoy, S.V. Solution method on the basis of rank approach for integer linear problems with boolean variables [Текст] / S.V. Listrovoy, D.Yu. Golubnichiy, E.S. Listrovaya // Engineering Simulation. – 1999. – Vol. 16. – P. 707–725.

105. Listrovoy, S.V. Parallel algorithms of calculation process optimization for the boolean programming problems [Текст] / S.V. Listrovoy, V.F. Tretjak, A.S. Listrovaya // Engineering Simulation. – 1999. – Vol. 16. – P. 569–579.

106. Листровой, С.В. Метод решения задач целочисленного линейного программирование с булевыми переменными на основе рангового подхода [Текст] / С.В. Листровой, Д.Ю. Голубничий, Е.С. Листровая // Электронное моделирование. – 1998. – Т. 20, № 6. – С. 14–32.

107. Листровой, С.В. Параллельный алгоритм для решения задачи линейного программирование с булевыми переменными [Текст] / С.В. Листровой // Электронное моделирование. – 1991. – Т. 13, № 3. – С. 29–32.

108. Листровой, С.В. Алгоритм решения задачи булевого линейного программирования в реальном масштабе времени [Текст] / С.В. Листровой, Д.Ю. Голубничий, А.Ю. Гуль // Обработка информации: сб. науч. трудов НАНУ. – Харьков: ПАНИ: ХВУ, 1994. – С. 56–59.

109. Третьяк, В.Ф. Алгоритм параллельных вычислений для решения задач ЦЛП с БП [Текст] / В.Ф. Третьяк, С.В. Листровой, С.В. Яблочков // Информационные системы: сб. науч. трудов НАНУ. – Харьков: НАНУ: ПАНИ: ХВУ, 1997. – Вып. 3 (11). – С. 54-57.

110. Листровой, С.В. Метод решения задачи о минимальном покрытии на основе рангового похода [Текст] / С.В. Листровой, А.Ю. Гуль // Электронное моделирование. – 1999. – № 1. – С. 58–70.

111. Листровой, С.В. Алгоритм точного решения задачи о наименьшем покрытии [Текст] / С.В. Листровой, А.Ю. Гуль, Г.В. Шубина // Модели и системы: науч. сб. – 1999. – Вып. 1. – С. 50–53.

112. Листровой, С.В. Точный алгоритм решения задачи о минимальном покрытии [Текст] / С.В. Листровой, А.Ю. Гуль, Е.С. Листровая // Информатика. – К.: Наук. думка, 1998. – Вып. 5. – С. 31–35.

113. Листровой, С.В. Алгоритм решения одномерной задачи (0,1)-рюкзак [Текст] / С.В. Листровой, Д.Ю. Голубничий // Информационные системы: сб. статей. – Харьков: НАНУ: ПАНИ: ХВУ, 1995. – С. 59–62.

114. Листровой, С.В. Алгоритм для идентификации изображений, работающий в масштабе реального времени [Текст] / С.В. Листровой, О.А. Дробот, О.И. Тимочко // Искусственный интеллект. – Донецк: Национальная академия наук Украина. Институт проблем искусственного интеллекта, 2003. – С. 173–178.

115. Баранов, В.Л. Моделирование многокритериальных задач целочисленного линейного программирования [Текст] / В.Л. Баранов, Г.Л. Баранов, Е.Ю. Комаренко // Электронное моделирование. – 1996. – Т. 18, № 6. – С. 15–22.

116. Баранов, В.Л. Моделирование дифференциальных игр на параллельных вычислительных структурах [Текст] / В.Л. Баранов // Моделирование-85: Теория, средства, применения. – К.: Институт проблем моделирования в энергетике АН Украины, 1985. – С. 18.

117. Баранов, В.Л. Модель дифференциальной игры с нулевой суммой в дискретном пространстве [Текст] / В.Л. Баранов // Электронное моделирование. – 1989. – № 4. – С. 28–35.

118. Подиновский, В.В. Математическая теория выработки решений в сложных ситуациях [Текст] / В.В. Подиновский. – М.: МО СССР, 1981. – 211 с.

119. Бережная, М.А. Отказоустойчивая система управления процессом выращивания крупногабаритных монокристаллов [Текст] / М.А. Бережная, В.С. Суздаль, И.И. Тавровский // Информатика и моделирование: Вестник НТУ “ХПИ”: сб. науч. трудов. – Харьков: НТУ “ХПИ”, 2007. – № 37. – С. 9–18.

120. Танаев, В.С. Декомпозиция и агрегирование в задачах математического программирования [Текст] / В.С. Танаев. – Минск: Наука и техника, 1987. – 183 с.

121. Танаев, В.С. Теория расписаний одностадийных систем [Текст] / В.С. Танаев, В.С. Гордон, Я.М. Шафранский. – М.: Наука, 1984. – 381 с.

122. Бережная, М.А. Тестовое диагностирование одномерных однородных структур [Текст] / М.А. Бережная, Я.Ю. Королева, М.Г. Рыжикова // Вестник НТУ “ХПИ”. – Харьков: НТУ “ХПИ”, 2008. – № 31. – С. 49–57.

123. Бережная, М.А. Синтез проверяющих тестов для однородных структур на основе циклических отличительных последовательностей [Текст] / М.А. Бережная, Я.Ю. Королева, М.Г. Рыжикова // Інформаційно – керуючі системи на залізничному транспорті. Перспективные компьютерно-управляемые и телекоммуникационные системы: материалы 21-й междунар. конф. (Алушта, 24-28 сентября 2008 г.). – Харків, 2008. – № 4 (додаток). – С. 29–33.

124. Бережная, М.А. Однородные вычислительные сети с реконфигурируемой структурой [Текст] / М.А. Бережная, Я.Ю. Королева // Технология приборостроения. – 2008. – №1. – С. 44–48.

125. Гуляницкий, Л.Ф. О методах дискретной оптимизации для многопроцессорных вычислительных комплексов [Текст] / Л.Ф. Гуляницкий, И.В. Сергиенко, А.Н. Ходзинский // Кибернетика. – 1988. – № 4. – С. 26–33.

126. Бережная, М.А. Характеристические последовательности в конечно-автоматных моделях дискретных устройств [Текст] / М.А. Бережная, Я.Ю. Королева // Вестник НТУ “ХПИ”. – Харьков: НТУ “ХПИ”, 2008. – № 56. – С. 19–25.

127. Васильев, В.В. Моделирование задач оптимизации и дифференциальных игр [Текст] / В.В. Васильев, В.Л. Баранов. – К.: Наука, 1989. – 296 с.

128. Бережная, М.А. Синхронизирующие последовательности в конечных детерминированных автоматах [Текст] / М.А. Бережная // Вестник НТУ “ХПИ”. – Харьков: НТУ “ХПИ”, 2008. – № 57. – С. 7–15.

129. Моисеев, Н.Н. Математические задачи системного анализа [Текст] / Н.Н. Моисеев. – М.: Наука, 1981. – 488 с.

130. Мирошник, М.А. Проектирование распределенных вычислительных систем на базе компьютерных сетей [Текст] / М.А. Мирошник // Інформаційно-керуючі системи на залізничному транспорті. – 2014. – № 6. – С. 3-7.

131. Листровой, С.В. О возможности распараллеливания комбинаторных задач на дереве путей графа [Текст] / С.В. Листровой, В.Я. Певнев // Вопросы оптимизации вычислений: всесоюзный семинар. докл. – К.: Институт кибернетики им. В.М. Глушкова АН УССР, 1987. – С. 185.

132. Листровой, С.В. Вопросы построения параллельных вычислительных систем и параллельный алгоритм для решения задачи о кратчайшем пути [Текст] / С.В. Листровой, В.Я. Певнев // Электронное моделирование. – 1990. – Т. 12, № 1. – С. 14–20.

133. Листровой, С.В. Параллельный алгоритм оптимальной раскраски графов [Текст] / С.В. Листровой, А.В. Сидоров // Обработка информации и обеспечение надежности систем управления: сб. науч. трудов НАНУ. – Харьков: ХВУ, 2004. – С. 136–139.

134. Дегтярев, Ю.И. Методы оптимизации [Текст] / Ю.И. Дегтярев. – М.: Советское радио, 1980. – 269 с.

135. Листровой, С.В. Архитектура параллельных вычислительных систем циклического типа [Текст] / С.В. Листровой // Электронное моделирование. – 1992. – Т. 14, № 2. – С. 28–36.

136. Листровой, С.В. Параллельный алгоритм определения кратчайшего остова в произвольном графе / С.В. Листровой // Материалы семинара Ин-т кибернетики АН УССР им. В.М. Глушкова. – Харьков: ХВВКИУ РВ. МО СССР, 1991. – С. 33.

137. Листровой, С.В. Многопроцессорная распределенная система управления [Текст] / С.В. Листровой // Тезисы докладов ин-та кибернетики

АН УССР В.М. Глушкова. – К.: Ин-т. кибернетики АН УССР В.М. Глушкова, 1988. – С. 65.

138. Листровой, С.В. Вопросы построения вычислительной системы для реализации параллельного алгоритма нахождения кратчайших путей [Текст] / С.В. Листровой, В.Я. Певнев // ЦИВТИ. – 1988. – Вып. 8. – С. 19.

139. Листровой, С.В. Недетерминированный параллельный алгоритм для решения NP-полных задач [Текст] / С.В. Листровой, В.Я. Певнев // ЦИВТИ. – 1989. – Вып. 8. – С. 21.

140. Листровой, С.В. Недетерминированные алгоритмы и NP-полные задачи [Текст] / С.В. Листровой, В.Я. Певнев // Тезисы докладов научно-тематической конференции вида РВА войск в/ч 25840. – 1989. – С. 3.

141. Листровой, С.В. Параллельный алгоритм для задачи о кратчайших маршрутах на графе [Текст] / С.В. Листровой // Техническая кибернетика: Изв. АН СССР. – 1990. – № 4. – С. 61–70.

142. Листровой, С.В. Метод определения путей с максимальной пропускной способностью [Текст] / С.В. Листровой, В.Н. Хрин. – Харьков: НАНУ: ПАНИ: ХВУ, 1996. – С. 59–62.

143. Листровой, С.В. Исследование вычислительной сложности алгоритма нахождения пути с максимальной пропускной способностью [Текст] / С.В. Листровой, В.Н. Хрин, Д.М. Вешкин // Обработка информации и обеспечение надежности систем управления: сб. науч. трудов НАНУ ПАНИ. – Харьков: ХВУ, 1997. – С. 137–139.

144. Листровой, С.В. Использование рангового подхода для решения задачи о вершинном покрытии [Текст] / С.В. Листровой, В.Н. Хрин, А.Ю. Гуль // Информационные системы: сб. науч. трудов НАНУ, ПАНИ. – Харьков: ХВУ, 1997. – Вып. 1. – С. 8–14.

145. Листровой, С.В. Параллельный алгоритм определения изоморфизма графов [Текст] / С.В. Листровой, В.Я. Певнев, А.Г. Тяжлов // Радиоэлектроника и информатика. – 1998. – № 2 (03). – С. 111–113.

146. Листровой, С.В. Параллельный алгоритм определения путей с максимальной пропускной способностью [Текст] / С.В. Листровой, В.Н. Хрин // Кибернетика и системный анализ. – 1998. – № 2. – С. 125–134.

147. Листровой, С.В. Параллельная реализация задач целочисленного линейного программирования на систолических однородных средах [Текст] / С.В. Листровой, С.В. Мищенко, В.Ф. Третьяк // Информационные технологии: наука, техника, образование: труды науч.-техн. конф. – Харьков: Гос. политех. ун-т: Мишкольц ун-т: Магдебург. ун-т, 1997. – С. 81–84.

148. Королев, А.В. Оптимальное планирование реализации алгоритмов в вычислительных системах [Текст] / А.В. Королев, О.М. Крючков, С.В. Листровой, В.Ф. Третьяк // Промышленность стройматериалов, энергоресурса сбережения в условиях рыночних отношений. Ч. 8. Математическое моделирование и информационные технологии: труды междунар. конф. – Белгород, 1997. – С. 163–171.

149. Листровой, С.В. Метод решения задач «вершинное покрытие» и «независимое множество» [Текст] / С.В. Листровой, С.В. Яблочков // Системи обробки інформації НАНУ, ПАНІ. – Харків: ХВУ, 2001. – Вип. 21. – С. 136–141.

150. Листровой, С.В. Алгоритмы решения задачи «3-выполнимость» [Текст] / С.В. Листровой, Е.С. Листровая, С.В. Яблочков // Вестник НТУ «ХПИ». – 2001. – Вып. 4. – С. 146–151.

151. Listrovoy, S.V. Method of Minimum Covering Problem Solution on the Basis of Rank Approach [Текст] / S.V. Listrovoy, A.Yu. Gul // Engineering Simulation. – 1999. – Vol. 17. –P. 73–89.

152. Листровой, С.В. Метод решения задачи 3-выполнимость [Текст] / С.В. Листровой // Электронное моделирование. – 2001. – № 6. – С. 66–76.

153. Листровой, С.В. Метод решения задачи определения минимальных вершинных покрытий и независимых максимальных множеств [Текст] / С.В. Листровой, С.В. Яблочков // Электронное моделирование. – 2003. – Т. 25, № 2. – С. 31–40.

154. А.с. 1462352 (СССР), МКИ 06F15/20. Устройство для определения путей в графе [Текст] / Листровой С.В. и др. (СССР). – № 4306139; заявл. 04.08.87; опубл. 28.02.89, Бюл. № 8.– 6 с.

155. А.с. 17658332 (СССР), МКИ 06F15/20. Устройство для решения задач на графах [Текст] / Листровой С.В. и др. (СССР). – № 4729168; заявл. 09.08.89; опубл. 30.09.92, Бюл. № 36. – 3 с.

156. А.с. 1832310 (СССР), МКИ 06F15/20. Устройство для решения задач на графах [Текст] / Листровой С.В. и др. (СССР). – № 4786697 (СССР); заявл. 30.01.90; опубл. 07.08.90, Бюл. № 29. – 4 с.

157. А.с. №1639303 (СССР), МКИ 06F15/20. Устройство для решения задач на графах [Текст] / Листровой С.В. и др. (СССР). – № 4474346 (СССР); заявл. 16.08. 88; опубл. 01.12.90, Бюл. №72.– 2 с.

158. А.с. 1705841 (СССР), МКИ 06F15/20. Устройство для решения задач на графах [Текст] / Листровой С.В. и др. (СССР). – № 4828057; заявл. 05.03.90; опубл. 15.01.92, Бюл. № 2. – 7 с.

159. А.с. 2478401 (СССР), МКИ 06F15/20. Устройство для решения задач на графах [Текст] / Листровой С.В. и др. (СССР). – № 4784401; заявл. 18.01.90; опубл. 30.08.93, Бюл. № 32. – 10 с.

160. А.С. №1639303 (СССР), МКИ 06F15/20. Устройство для решения задач на графах [Текст] / Листровой С.В. и др. (СССР); – № 4474346; заявл. 16.08.88; опубл. 01.12.90, Бюл. № 72.– 2с.

161. Бережная, М.А. Синтез проверяющих тестов для сетей клеточных автоматов с наблюдаемыми выходами [Текст] / М.А. Бережная, Я.Ю. Королева // Технология приборостроения. – 2008. – № 2. – С. 20–24.

162. Гери, М. Вычислительные машины и труднорешаемые задачи [Текст] / М. Гери, Д. Джонсон. – М.: Мир, 1982.– 416 с.

163. Емеличев, В.А. Лекции по теории графов [Текст] / В.А. Емеличев, О.И. Мельников, В.И. Сарванов, Р.И. Тишкевич. – М.: Наука, 1990. – 382 с.

164. Зиков, А.А. Основы теории графов [Текст] / А.А. Зиков. – М.: Наука, 1987.– 380 с.

165. Свами, М. Графы, сети и алгоритмы [Текст] / М. Свами, К. Тхуласираман. – М.: Мир, 1984.–450 с.

166. Хохлюк, В.И. О параллелизме в целочисленной оптимизации [Текст] / В.И. Хохлюк // Вычислительные системы. – Новосибирск: ИН СО АН СССР, 1981. – Вып. 84. – 93 с.

167. Пристрій регулювання росту монокристалів та пристрій діагностування для нього [Текст]: пат. 86105 Україна: МПК[7] С30 В 15/20, G06 F 11/28 / М.А. Бережна, Л.В. Дербунович, Я.Ю. Корольова, В.С. Суздаль; заявник і патентовласник Інститут сцинтиляційних матеріалів НАН України. – №200808602; заявл. 01.07.08; опубл. 25.03.2009, Бюл. №1. – 25 с.

168. Бережная, М.А. Применение сетей клеточных автоматов в криптографических системах / М.А. Бережная, Я.Ю Королева // Методи та засоби кодування, захисту й ущільнення інформації: ІІ-га міжнар. наук.-практ. конф. (Вінниця, 22-24 квітня 2009 р.). – Вінниця, 2009. – С. 94–96.

169. Васильев, В.В. Электронные модели задач на графах [Текст] / В.В. Васильев, Е.А. Ралдугин. – К.: Наук. думка, 1987. – 152 с.

170. Бережная, М.А. Микропроцессорные системы управления с реконфигурируемой структурой [Текст] / М.А. Бережная, Я.Ю. Королева // Технология приборостроения. – 2009. – № 2. – С. 16–23.

171. Бережная, М.А. Диагностические эксперименты в системах защиты информации на сетях клеточных автоматов [Текст] / М.А. Бережная, Я.Ю. Королева // Інформаційно – керуючі системи на залізничному транспорті. – 2009. – № 4. – С. 142–145.

172. Белколин, В.Е. Анализ и синтез систем автоматизированного управления на ЭВМ. Алгоритмы и программы [Текст] / В.Е. Белколин, П.И. Чинаев. – М.: Радио и связь, 1986. – 248 с.

173. Бережная, М.А. Синтез проверяющих тестов для однородных схем [Текст] / М.А. Бережная, Я.Ю. Королева // Біоніка інтелекту. – 2009. – № 4. – С. 142–145.

174. Бережная, М.А. Методы проектирования нечетких устройств принятия решений на основе программируемых логических интегральных микросхем [Текст] / М.А. Бережная, Я.Ю. Королева // Технология приборостроения. – 2009. – № 2. – С. 16–23.

175. Байков, В.Д. Специализированные процессоры: итерационные алгоритмы и структуры [Текст] / В.Д. Байков, В.Б. Смолов. – М.: Радио и связь, 1985. – 288 с.

176. Венцель, Е.С. Исследование операций. Задачи, принципы, методология [Текст] / Е.С. Венцель. – М.: Наука, 1980.– 208 с.

177. Исследование операций [Текст]: пер. с англ. / под ред. Дж. Моудера, С. Елмаграби. – М.: Мир, 1981. – Т. 2. – 677 с.

178. Филлипс, Д. Методы анализа сетей [Текст]: пер. с англ. / Д. Филлипс, А. Гарсиа-Диас. – М.: Мир, 1984.– 496 с.

179. Рощин, В.А. О решении задач целочисленного программирования с неоднозначно заданными данными [Текст] / В.А. Рощин, И.В. Сергиенко // Кибернетика и системный анализ. – 1994.– № 5.– С. 45–51.

180. Михалевич, В.С. К вопросу оптимизации вычислений [Текст] / В.С. Михалевич, И.В. Сергиенко // Кибернетика и системный анализ. – 1994.– № 2.– С. 65–93.

181. Баранов, В.Л. Рефлексивные модели конфликтного взаимодействия автоматизированных комплексов радиоэлектронной борьбы и системы военной радиосвязи [Текст] / В.Л. Баранов // Электронное моделирование. – 1995. – Т. 17, № 4. – С. 57–64.

182. Баранов, В.Л. Моделирование многокритериальных задач управления методом дифференциальных преобразований [Текст] / В.Л. Баранов // Электронное моделирование. – 1995.– Т. 17, № 1.– С. 16–23.

183. Листровой, С.В. Параллельные комбинаторные алгоритмы [Текст] / С.В. Листровой, В.Я. Певнев // Формальные модели параллельных вычислений: тезисы докладов междунар. науч. конф. – Новосибирск, 1987. – С. 4.

184. Листровой, С.В. Многопроцессорная распределенная система управления [Текст] / С.В. Листровой, В.Я. Певнев // Проектирование автоматизированных систем контроля и управление сложными объектами: тезисы докладов междунар. школы. – Туапсе, 1988. – С. 42.

185. Листровой, С.В. Об использовании гарантированных прогнозов в методах решения задач булевого программирования на основе рангового подхода [Текст] / С.В. Листровой, О.Н. Симашкевич // Электронное моделирование. – 2003. – Т. 25, № 4. – С. 89–103.

186. Тельпиз, М.И. Представления функций алгебры логики [Текст] / М.И. Тельпиз // Кибернетика. – № 4. – С. 37–51.

187. Тельпиз, М.И. Позиционные операторы и преобразования в алгебре логики [Текст] / М.И. Тельпиз. – М.: Кибернетика, 1985. – 60 с. – (Препринт / АН СССР Научный совет по комплексной проблеме «Кибернетика»).

188. Тельпиз, М.И. Алгебра позиционных операторов и еквивалентных преобразований [Текст] / М.И. Тельпиз. – М.: Кибернетика, 1988.– 64 с. – (Препринт / АН СССР Научный совет по комплексной проблеме «Кибернетика»).

189. Тельпиз, М.И. Принцип позиционности в трехзначной алгебре логики [Текст] / М.И. Тельпиз, С.М. Демидчик, Л.М. Щербанский. – М.: Кибернетика, 1988. – 20 с. – (Препринт / АН СССР Научный совет по комплексной проблеме «Кибернетика»).

190. Тельпиз, М.И. Позиционние фундаментальние симметрические операторы и задачи логического распознавания логики / М.И. Тельпиз. – М.: Кибернетика, 1989.– 72 с. – (Препринт / АН СССР Научный совет по комплексной проблеме «Кибернетика»).

191. Липский, В. Комбинаторика для программистов [Текст] / В. Липский. – М.: Мир, 1988. – 203 с.

192. Шор, Н.З. Квадратичные экстремальные задачи и недифференцируемая оптимизация [Текст] / Н.З. Шор, С.И. Стеценко. – К.: Наук. думка, 1989. – 196 с.

193. Любченко, Г.Г. Реферативный журнал математика 8В321 [Текст] / Г.Г. Любченко, В.С. Подлипенский. – М., 1980. – 95 с.

194. Пристрій для діагностування пристрою регулювання росту монокристалів [Текст]: пат. 89312 Україна, МПК[8] G06 F 11/28, C30 B 15/20 / М.А. Бережна, Л.В. Дербунович, Я.Ю. Корольова, В.С. Суздаль; заявник і патентовласник Інститут сцинтиляційних матеріалів НАН України. – № 200808602; заявл. 01.07.08; опубл. 11.01.2010, Бюл. №1. – 23 с.

195. Бережная, М.А. Диагностическая инфраструктура с интеллектуальными свойствами в реконфигурируемых мультипроцессорных системах [Текст] / М.А. Бережная, Л.В. Дербунович, Я.Ю. Королева // Інформаційні технології та компютерна інженерія: міжнар. наук.-практ. конф. (Вінниця, 19-21 травня 2010 р.). – Вінниця, 2010. – С. 344–346.

196. Юдин, Д.Б. Число и мысль [Текст] / Д.Б. Юдин, А.Д. Юдин. – М.: Знание, 1985. – Вып. 8. – С. 189.

197. Успенский, В.А. Теорема Геделя о неполноте [Текст] / В.А. Успенский – М.: Наука, глав. редакция физ.-мат. лит., 1982. – 111 с.

198. Успенский, В.А. Теория алгоритмов. Основные открытия и приложения [Текст] / В.А. Успенский, А.Л. Семенов. – М.: Наука, глав. редакция физ.-мат. лит., 1987. – 288 с.

199. Виленкин, Н.Я. В поисках бесконечности [Текст] / Н.Я. Виленкин // М.: Наука, 1983. – 160 с.

200. Кузнецов, О.П. Дискретная математика для инженера [Текст] / О.П. Кузнецов, Г.М. Адельсон-Вельский. – М.: Энергия, 1980. – 342 с.

201. Кожухов, В.Д. Определение выполнимости расписаний при управлении сложными системами [Текст] / В.Д. Кожухов, Е.С. Листровая, Д.М. Вешкин // Вісті академії інженерних наук. – 1998. – С. 103–107.

202. Городнов, В.П. Моделирование боевых действий частей, соединений и объединений войск ПВО [Текст] / В.П. Городнов. – Харьков: ВИРТА ПВО, 1987. – 380 с.

203. Ловас, Л. Прикладные задачи теории графов [Текст]: науч.-практ. издание / Л. Ловас, М. Пламер. – М.: МИР, 1998.– 645 с.

204. Оре, О. Теория графов [Текст] / О. Оре. – М.: Наука, 1980.– 336 с.

205. Кожухов, В.Д. Методи рішення задачі про найменший поділ на основі рангового підходу [Текст] / В.Д. Кожухов, О.С. Лістрова // Вісті

академії інженерних наук. – 1999. – № 4. – С. 117–123.

206. Корнеев, А.И. Метод решения задачи ЦЛП на основе рангового подхода [Текст] / А.И. Корнеев, Е.С. Листровая // Модели и системы: сб. ХВУ. – 1999. – № 1. – С. 40–44.

207. Гиневский, М.И. Влияние сортировок вершин в графе n-мерного единичного куба на погрешность при решении задачи о взвешенном минимальном покрытии [Текст] / М.И. Гиневский, Е.С. Листровая, Д.С. Пивнев // Системи обробки інформації: зб. наук. праць ХВУ. – 2000. – Вип. 4 (10). – С. 172–174.

208. Бережна, М.А. Метод синхронного тестування дискретних пристроїв [Текст] / М.А. Бережна [та ін.] // Інформаційні технології: наука, техніка, технологія, освіта, здоров'я: XII-та міжнар. наук.-практ. конф. Секція 8. Мікропроцесорна техніка в автоматиці й приладобудуванні (Харків, 20-21 травня, 2004 р.) – Харків, 2004. – С. 27–28.

209. Мирошник, М.А. Диагностирование микропроцессорных систем на системном и функциональном уровне [Текст] / М.А. Мирошник // Інформаційно – керуючі системи на залізничному транспорті. – 2010. – №3. – С. 3–41.

210. Мирошник, М.А. Разработка диагностических автоматных моделей динамических систем [Текст] / М.А. Мирошник, Я.Ю. Королева, Н.О. Замирец // Технология приборостроения. – 2010. – №1. – С. 15–18.

211. Garey, M. Computers and hard-solved problems [Текст] / M. Garey, Johnson D Original // Computers and Intractability. A Guide to the Theory of NP-completeness // Translation from English. – M.: Mir, 1982. – 416 p.

212. Telpiz, M.I. Representations of functions of algebra of logic [Текст] / M.I. Telpiz // Cybernetics. – K., 1985. – № 4.– P. 37–51.

213. Telpiz, M.I. Algebra of positional operators and equivalent transformations. Preprint [Текст]: Academy of sciences of the USSR / M.I. Telpiz // Scientific advice on a complex problem "Cybernetics". – M., 1988. – 64 p.

214. Telpiz, M.I. Chtcherbanski L.M. Positionality principle in three-value algebra of logic [Текст]: Preprint – 1436 / M.I. Telpiz, S.M. Demidchik // Institute of space researches of the Russian Academy of Science. – M., 1988. – 20 p.

215. Telpiz, M.I. Positional fundamental symmetric operators and problems of logic recognition [Текст]: Preprint – 1601 / M.I. Telpiz // Institute of space researches of the Russian Academy of Science. – M., 1989.– 72 p.

216. Kinosita, K. Logic designing of the superbig integrated circuits [Текст]: Translation from Japanese / K. Kinosita. – M.: Mir, 1988. – 309 p.

217. Kung, S. Matrix processors on the superbig integrated circuit [Текст]: Original: VLSI Array Processors: Translation from English / S. Kung. – M.: Mir, 1991. – 672 p.

218. Logic programming [Текст]: Translation from English and French. – M.: Mir, 1988. – 368 p.

219. Lauriere, J.L. Systems of an artificial intellect [Текст]: Original: Intelligence Artificielle: Translation from French / J.L. Lauriere. – M.: Mir, 1991. – 568 p.

220. Sterling, L. Art of programming in language the Prolog [Текст]: Original: The art of Prolog: Translation from English / L. Sterling, E. Shapiro. – M.: Mir, 1990. – 235 p.

221. Waterman, M.S. Mathematical methods for DNA sequences [Текст]: Translation from English / Editor M.S.Waterman. – M.: Mir, 1999. – 349 p.

222. Сергиенко, И.В. Об основных направлениях развития информатики [Текст] / И.В. Сергиенко // Кибернетика и системный анализ. – 1997. – № 6. – С. 3–33.

223. Мирошник, М.А. Синтез легкотестируемых двумерных сетей клеточных автоматов [Текст] / М.А. Мирошник, Я.Ю. Королева // Информационно-управляющие системы на железнодорожном транспорте. Перспективные технологии компьютерных управляющих и телекоммуникационных систем: XXIII-я междунар. конф. – 2010. – № 4 (додаток) – С. 69–73.

224. Капитонова, Ю.В. Системное математическое обеспечение многопроцессорного вычислительного комплекса ЕС [Текст] / под ред. Ю.В. Капитоновой. – М.: ВВИА им. проф. Н.Е. Жуковского, 1986. – 350 с.

225. Тарков, М.С. О решении на вычислительных системах с программируемой структурой трехдиагональных систем уравнений параметрическим методом [Текст] / М.С. Тарков // Электронное моделирование. – 1990.– Т. 12, № 5.–С. 8–12.

226. Мелентьев, В.А. Моделирование параллельной обработки изображений на базе вычислительной системы МИКРОС-2 [Текст] / В.А. Мелентьев, М.С. Тарков, Н.Г. Грязнов и др. // Электронное моделирование. – 1995.– Т. 17, № 2. – С. 49–55.

227. Тарков, М.С. Организация отказоустойчивых параллельных вычислений в транспьютерной системе обработки изображений [Текст] / М.С. Тарков // Научное приборостроение. – 1995.– № 3. – С. 74–80.

228. Tarkov, M.S. Mapping parallel programs onto distributed robust computer systems [Текст] / M.S. Tarkov // Proc. Of 15th IMACS World Congress on Scientific Computation / Modelling and Applied Mathematics, Berlin, August, , Application in Modelling and Simulation Ed. by Achim Sydow. – 1997. – Vol. 6. – P. 365–370.

229. Тарков, М.С. Параллельный алгоритм бисекции структуры распределенной вычислительной системы [Текст] / М.С. Тарков // Распределенная обработка информации: труды VI-го Междунар. семинара (Академгородок, Новосибирск, 23-25 июня 1998 г.). – Новосибирск, 1998. – С. 96–100.

230. Тарков, М.С. Организация межпроцессных взаимодействий в транспьютерной системе МИКРОС-Т [Текст] / М.С. Тарков, В.А.

Чумаков // Распределенная обработка информации: труды VI-го Междунар. семинара – (Академгородок, Новосибирск, 23-25 июня 1998 г.). – Новосибирск, 1998. – С. 155–159.

231. Тарков, М.С. Самостабилизация покрывающего дерева в макроструктуре распределенной вычислительной системы [Текст] / М.С. Тарков // Автометрия. – 2001. – № 2. – С. 66–78.

232. Tarkov, M.S. Data Input Algorithm for Image Processing in a Distributed Computer System [Текст] /M.S. Tarkov, A.V. Tsarenko // Pattern Recognition and Image Analysis. MAIK Nauka. Interperiodica (Russia). – 2001.– Vol. 11, №. 2.– P. 381–383.

233. Tarkov, M.S. Mapping Adaptive Fuzzy Kohonen Clustering Network onto Distributed Image Processing System [Текст] / M.S. Tarkov, Y. Mun, J. Choi, H.I. Choi // Parallel Computing. – 2002. – Vol. 28, № 9. – P. 1239-1256.

234. Тарков, М.С. Вложение структур параллельных программ в структуры живучих распределенных вычислительных систем [Текст] / М.С. Тарков // Автометрия. – 2003.– № 3. –С. 84-96.

235. Йордан, Е. Структурные модели в объектно-ориентированном анализе и проектировании [Текст]: пер. с англ. / Е. Йордан, К. Аргила. – М.: Лори, 1999.– 233 с.

236. Калянов, Г.Н. CASE структурные модели в объектно-ориентированном анализе и проектировании [Текст]: пер. с англ. / Г.Н. Калянов. – М.: Лори, 1999. – 220 с.

237. Корнеев, В.В. Параллельные вычислительные системы [Текст] / В.В. Корнеев. – М.: Нолидж, 1999. – 245 с.

238. Тербер, К.Дж. Архитектура высокопроизводительных вычислительных систем [Текст] / К. Дж. Тербер. – М.: Наука, 1985.– 133 с.

239. Родрига, Г. Параллельные вычисления [Текст] / Г. Родрига. – М.: Наука, 1986. – 376 с.

240. Бернацкий, Ф.И. Построение области робастных управлений на параллельных процессорах [Текст] / Ф.И. Бернацкий, Г.Б. Диго, Н.Б. Диго // Информатика и системы управления. – 2003. – № 1 (5). – С. 92–100.

241. Кормен Т. Алгоритмы, построение и анализ [Текст] / Т. Кормен, Ч. Лейзерсон, Р. Ривест. – М.: МЦНМО, 2002. – 955 с.

242. Аноприенко, А.Я. Применение методов параллельного программирования в интегрированной навигационной системе для судов внутреннего и смешанного плавания [Текст] / А.Я. Аноприенко, С.В. Кривошеев // Интеллектуальные многопроцессорные системы: тезисы докладов междунар. науч.-техн. конф. (Таганрог, 1-5 сентября 1999 г.). – Таганрог, 1999. – С. 75–77.

243. Петренко, А.В. Модельные исследования различных архитектур кеш-памяти [Текст] / А.В. Петренко, А.Я. Аноприенко // Машиностроение и техносфера на рубеже XXI века: сб. трудов VI-й междунар. конф. (Севастополь, 13–18 сентября 1999 г.). – Севастополь, 1999. –С. 249–252.

244. Бернацкий, Ф.И. Распараллеливание вычислений при построении областей допустимих управлений [Текст] / Ф.И. Бернацкий, Г.Б. Диго, Н.Б. Диго // Надежность и качество: труды Междунар. симпозиума. – Пенза: ПГУ, 2003. – С. 24–26.

245. Beynon-Davies, P. Systemy baz danych [Текст] / P. Beynon-Davies // Wydannie drugie. Wydawnictwo Naukowo-Techniczne. – Warszawa, 2000.– 201 p.

246. Бережная М.А. Однородные сети с распределенной системной реконфигурацией [Текст] / М.А. Бережная, Л.В. Дербунович, Я.Ю. Королева // Вестник НТУ “ХПИ”. – Харьков: НТУ “ХПИ”, 2010. – № 20. – С. 71–78.

247. Дейт, К. Введение в системы баз даных [Текст]: пер. с англ. / К. Дейт. – 6-е изд. – К.; М.; СПб.: Издательский дом «Вильямс», 2000. – 848 с.

248. Хаббард, Ю.Дж. Автоматизированное проектирование баз данных [Текст] / Ю.Дж. Хаббард. – М.: Мир, 1984. – 294 с.

249. Мирошник, М.А. Разработка диагностического обеспечения многопроцессорных телекоммуникационных систем управления на основе концепции сигнатурного мониторинга [Текст] / Г.И. Загарий, М.А. Мирошник, С.В. Панченко // Інформаційно – керуючі системи на залізничному транспорті. – 2011. – №1. – С. 37–46.

250. Андрианов, А.Н. Язык Норма [Текст] / А.Н. Андрианов, К.Н. Ефимкин, И.Б. Задихайло, Н.В. Поддерюгина. – 1985. – №165. – 34 с. – (Препринт ИПМ им. Келдиша М.В. АН СССР).

251. Ивенс, Д. Системы параллельной обработки [Текст] / Д. Ивенс. – М.: Мир, 1985.– 416 с.

252. Котов, В.Е. Элементы параллельного программирования [Текст] / В.Е. Котов.–М.: Радио и связь, 1983. – 240 с.

253. Феррари, Д. Оценка производительности вычислительных систем [Текст]: пер. с англ. / под ред В.В. Мартинюка. – М.: Мир, 1981. – 351 с.

254. Андрианов, А.Н. Непроцедурный язык Норма и методы его реализации [Текст] / А.Н. Андрианов, К.Н. Ефимкин, И.Б. Задихайло // Языки и параллельные ЭВМ. Серия «Алгоритмы и алгоритмические языки». – М.: Наука, 1990. – С. 3–37.

255. Бебб, Р. Программирование на параллельных вычислительных системах [Текст] / Р. Бебб, Дж. Мак-Гроу, Т. Акселрод; под ред. Р. Бебба. – М.: Мир, 1991. – 376 с.

256. Абрамов, О.В. Об использовании технологии параллельных вычислений в задачах анализа и оптимизации параметрической надежности [Текст] / О.В. Абрамов, Я.В. Катуева // Многопроцессорные и вычислительные системы. II-я междунар. конф. по проблемам управления: тезисы докладов. – М.: Институт проблем управления, 2003. – Т. 2. – С. 140.

257. Бугеря, А.Б. Реализация математических функций языка Норма для распределенных вычислительных систем [Текст] / А.Б. Бугеря. – М.: ИПМ, 1994. – № 37. – 26 с. – (Препринт ИПМ им. Келдиша М.В. РАН).

258. Andrianov, A.N. Nonprocedural language Norma and problems of its implementation [Текст] / A.N. Andrianov, K.N. Efimkin, I.B. Zadykhailo // Proc. First Int. Workshop "Software for Multiprocessors and Supercomputers: Theory, Practice, Experience" (Moskow, Russia, September 19-23, 1994) – M., 1994. – P. 523–530.

259. Педдок, Р. Visual FoxPro 6. Разработка корпоративных приложений [Текст]: пер. с англ / Р. Педдок, Дж. Петерсен, Р. Телмейдж, Е. Ренфт. – М.: ДМК, 1999. – 592 с.

260. Андрианов, А.Н. Норма. Описание языка. Рабочий стандарт / А.Н. Андрианов, А.Б. Бугеря, К.Н. Ефимкин, И.Б. Задихайло. – 1995. – № 120.– 50 с. – (Препринт ИПМ им. Келдиша М.В. РАН).

261. Задихайло, И.Б. Содержательные обозначения и языки нового поколения [Текст] / И.Б. Задихайло, К.Н. Ефимкин // Информационные технологии и вычислительные системы. – 1996. – № 2. – С. 46–58.

262. Андрианов, А.Н. Организация циклического процесса по непроцедурной записи [Текст] / А.Н. Андрианов, Е.А. Андрианова // Программирование. – 1996.– № 4. – С. 62–72.

263. Андрианов, А.Н. Решение двумерных задач газовой динамики методом Годунова на параллельных ЭВМ с помощью языка Норма [Текст] / А.Н. Андрианов, С.Б. Базаров, К.Н. Ефимкин. – М.: ИПМ, 1997. – № 9. – 23 с. – (Препринт ИПМ им. Келдиша М.В. РАН).

264. Андрианов, А.Н. Применение языка Норма для решения трехмерных задач газовой динамики методом Годунова на многопроцессорных ЭВМ [Текст] / А.Н. Андрианов, С.Б. Базаров, К.Н. Ефимкин. – М.: ИПМ, 1997. – № 63.– 17 с. – (Препринт ИПМ им. Келдиша М.В. РАН).

265. Андрианов, А.Н. Введение в язык Норма и опыт его применения для решения задач пользователей [Текст] / А.Н. Андрианов, А.Б. Бугеря, К.Н. Ефимкин // Фундаментальные и прикладные аспекты разработки больших распределенных программных комплексов: тезисы Всеросс. науч. конф. – Новороссийск, 1998. – С. 11–15.

266. Ефимкин, К.Н. Язык Норма и методы его трансляции для параллельных вычислительных систем [Текст] / К.Н. Ефимкин // Фундаментальные и прикладные аспекты разработки больших распределенных программних комплексов: тезисы Всеросс. науч. конф. – Новороссийск, 1998. – С. 117–120.

267. Андрианов, А.Н. Использование языка Норма для решения вычислительных задач на нерегулярных сетках [Текст] / А.Н. Андрианов // Фундаментальные и прикладные аспекты разработки больших распределенных программных комплексов: тезисы Всеросс. науч. конф. – Новороссийск, 1998. – С. 120–123.

268. Andrianov, A.N. The declarative Norma language for the solution of mathematical physics problems on multiprocessors computers [Текст] / A.N. Andrianov, S.B. Bazarov, K.N. Efimkin, I.B. Zadykhailo // Proc. Int. Congress of Mathematicians, Berlin, August 18–27, 1998. – С. 115.

269. Мирошник, М.А. Методы защиты информации в распределенных компьютерных сетях [Текст] / М.А. Мирошник // Інформаційно-керуючі системи на залізничному транспорті. – 2014. – № 5. – С. 66-70.

270. Ортега, Дж. Введение в параллельные и векторные методы решения линейных систем [Текст] / Дж. Ортега. – М.: Мир, 1991. – 256 с.

271. Хоар, Ч. Взаимодействие последовательных процессов [Текст] / Ч. Хоар. – М.: Мир, 1989.–264 с.

272. Джексон, Г. Проектирование реляционных баз данных для использования с микроЭВМ [Текст]: пер. с англ / Г. Джексон. – М.: Мир, 1991. – 252 с.

273. Хансен, Г. Базы данных: разработка и приложение [Текст]: пер. с англ. / Г. Хансен, Дж. Хансен. – М.: БИНОМ, 1999. – 704 с.

274. Сосински Б. Разработка приложений в среде Visual FoxPro 5 [Текст]: пер. с англ. / Б. Сосински. – К.: Диалектика, 1997.– 448 с.

275. Андрианов, А.Н. Использование системы НОРМА для решения вычислительных задач на многопроцессорных системах с распределенной памятью [Текст] / А.Н. Андрианов, К.Н. Ефимкин // Вычислительные методы и программирование. – 2000.– Т. 1. – С. 45–54.

276. Мирошник, М.А. Размещение подзадач в распределенных вычислительных системах кластерно-метакомпьютерного типа [Текст] / М.А. Мирошник, Л.А. Клименко // Інформаційно-керуючі системи на залізничному транспорті. – 2014. – №4. – С.71-77.

277. Мирошник, М.А. Методы интерактивного управления ресурсами технических систем при проектировании. [Текст] / М.А. Мирошник, Ю.Н. Салфетникова. // Інформаційно-керуючі системи на залізничному транспорті. Перспективні компютерні управляючі и телекомунікаційні системи: XXVII-та міжнар. конф. (Алушта, 24-26 вересня 2014). – Харків, 2014. – №4 (додаток). – С. 4.

278. Каратигин, С.А. Visual FoxPro 6 [Текст] / С.А. Каратигин, А.Ф. Тихонов, Л.Н. Тихонова. – М.: ЗАО "Издательство БИНОМ", 1999. – 784 с.

279. Баженова, И.Ю. Visual FoxPro 6 [Текст] / И.Ю. Баженова.– М.: Диалог-МИФИ, 1999. – 416с.

280. Miroschnik, M.A. Investigation of the influence of generator signal higher order propagation components and ways of its compensating in the multiprobe microwave multimeter [Текст] / M.A. Miroschnik, O. B. Zaichenko, M.A. Kovalenko // Інформаційно-керуючі системи на залізничному транспорті. – 2014. – №3. – С. 78-83.

281. Andrianov, A.N. The Norma language application to solution of strong nonequilibrium transfer process problem with condensation of mixtures on the multiprocessor system [Текст] / A.N. Andrianov, K.N. Efimkin, V.Y. Levashov, I.N. Shishkova // Computational Science – ICCS 2001, Lecture Notes in Computer Science. – May 2001. – Vol. 2073. – P. 502–510.

282. Мирошник, М.А. Синтез распределенных компьютерных сред на базе компьютерных сетей [Текст] / М.А. Мирошник // Системи обробки інформації. – 2013. – № 7 (114). – С. 86-89.

283. Мирошник, М.А. Разработка средств организации функционирования распределённых вычислительных систем и сетей [Текст] / М.А. Мирошник, Н.В. Мирошник // Інформаційно-керуючі системи на залізничному транспорті. Перспективні компютерні управляючі и телекоунікаційні системи: тези доповіді XXVI-ї міжнар. конф. (Алушта, 24-28 вересня 2013 р.). – Харків, 2013. – № 4 (додаток). – С. 11.

284. Miroschnik, M.A. Uses of programmable logic integrated circuits for implementations of data encryption standard and its experimental linear cryptanalysis [Текст] / M.A. Miroschnik, M.A. Kovalenko // Інформаційно-керуючі системи на залізничному транспорті. – 2013. – № 6. – С. 36-45.

285. Андрианов, А.Н. НОРМА – специализированная система параллельного программирования [Текст] / А.Н. Андрианов, А.Б. Бугеря, К.Н. Ефимкин // Сибирская школа-семинар по параллельному программированию (Томск, 17-20 декабря 2001 г.): сб. материалов. – Томск: Изд-во Томского университета, 2002. – С. 33–45.

286. Сигал, И.Х. Введение в прикладное дискретное программирование [Текст] / И.Х. Сигал, А.П. Иванова. – М.: ФИЗМАТЛИТ, 2002. – 240 с.

287. Новиков, Ф.А. Дискретная математика для программистов [Текст] / Ф.А. Новиков. – СПб., 2000. – 303 с.

288. Корнеев, В.Г. Параллельные вычислительные системы [Текст] / В.Г. Корнеев. – М.: Нолидж, 1999. – 320 с.

289. Самофалов, В.В. Информационно-вычислительная структура суперкомпьютерного вычислительного центра ИММ УрО РАН [Текст] / В.В. Самофалов, А.С. Игумнов, А.В. Коновалов, С.В. Шарф // Научный сервис в сети Интернет: тезисы докладов Всеросс. науч. конф. (Новороссийск, 18-23 сентября 2000 г.). – М.: Изд-во МГУ, 2000. – С. 133-135.

290. Абрамов, С.М. Проектирование высокопроизводительного процессора обработки графов [Текст] / С.М. Абрамов, А.В. Кондратьева, В.А. Роганов, А.М. Чеповский // Информационные технологии. – М.: Машиностроение, 2001. – № 3. – С. 34-39.

291. Айламазян, А.К. Данные, документы и архитектура медицинских информационных систем [Текст] / А.К. Айламазян, Я.И. Гулиев // Тезисы докладов Междунар. форума мед. лит. – 2001. – С. 41-50.

292. Khachumov, V.M. Frequency-Parametric Representation of Curves in Systems of Computer Graphics and Image Processing [Текст] / V.M. Khachumov // Pattern Recognition and Image Analysis. – 2001. – Vol. 11, № 2. – P. 440-441.

293. Гурман, В.И. Особые обобщения и оптимальные управления [Текст] / В.И. Гурман // Труды НАН Белоруссии. – Минск, 2001. – Т.7. – С. 43–52.

294. Бахвалов, Н.С. Численные методы в задачах и упражнениях [Текст] / Н.С. Бахвалов, А.В. Лапин, Е.В. Чижонков. – М.: Высшая школа, 2000. – 192 с.

295. Ломазова, И.А. Рекурсивные вложенные сети Петри: анализ семантических свойств и выразительность [Текст] / И.А. Ломазова // Программирование. – 2001. – № 4.–С. 21–35.

296. Lomazova, I.A. Nested Petri nets: multi level and recursive systems [Текст] / I.A. Lomazova // Fundamenta Informaticae. – 2001. – Vol. 47, № 3-4. – P. 283–293.

297. Абрамов, С.М. Динамическое распараллеливание программ на базе параллельной редукции графов. Архитектура программного обеспечения новой версии Т-системы [Текст] / С.М. Абрамов, В.А. Васенин, Е.Е. Мамчиц [и др.] // Научная сессия МИФИ: сб. науч. трудов (Москва, 22-26 января 2001 г.). – М., 2001. – Т. 2. –С. 234.

298. Абрамов, С.М. Проектирование высокопроизводительного процессора обработки графов [Текст] / С.М. Абрамов, А.В. Кондратьева, В.А. Роганов, А.М. Чеповский // Информационные технологии. – М.: Машиностроение, 2001. – № 3. – С. 12–21.

299. Ломазова, И.А. Анализ семантических свойств некоторых классов программ и сетей Петри [Текст]: автореф. дисс. … д-ра физ.-мат. наук: 05.13.17 / И.А. Ломазова. – М., 2001.– 32 с.

300. Бурдаев, М.Н. Принципы построения интеллектуальной измерительно-управляющей системы [Текст] / М.Н. Бурдаев [и др.] // Космос без оружия – арена мирного сотрудничества в XXI веке: тезисы докл. междунар. космич. конф. (Москва, апрель 2001 г.). – М.: МАИ, 2001. – С. 48.

301. Ахмедов, С.А. Использование гибридного подхода к представлению знаний на основе фреймов и семантических сетей для задач классификации [Текст] / С.А Ахмедов [и др.] // Информационные технологии в науке, образовании, телекоммуникации, бизнесе: труды XXVIII конф. IT+SE'2001 (Ялта, 20-29 мая 2001 г.). – Гурзуф, 2001. – С. 89–91.

302. Абрамов, С.М. Динамическое распараллеливание программ на базе параллельной редукции графов. Архитектура программного обеспечения новой версии Т-системы [Текст] / С.М. Абрамов [и др.] // Научая сессия МИФИ (Москва, 22-26 января 2001 г.): сб. науч. трудов. – М., 2001.–Т.2.– 234 с.

303. Kornev, A.A. On an unstable manifold and approximation attractor of a symidynamic system on a parallel computer under a T-system [Текст] / A.A. Kornev [и др.] // Department of mathematics university of Nijmegen The Netherlands: Report (01.04. 2001.). – P. 112–121.

304. Белишев, Д.В. Интеллектуальные процедуры оптимального управления [Текст] / Д.В. Белишев, В.И. Гурман // Автоматика и телемеханика.–2002. – № 5.– С. 147–155.

305. Белишев, Д.В. Реализация многометодных процедур оптимального управления [Текст] / Д.В. Белишев // Компьютерное и математическое моделирование в естественных и технических науках: материалы IV Всеросс. науч. internet-конф. (апрель-май 2002 г.). – 2002. – № 16. – С. 3–7.

306. Белишев, Д.В. Мультиметодные процедуры оптимального управления [Текст] / Д.В. Белишев [и др.] // Обобщенные решения в задачах управления: труды междунар. симпозиума: тезисы докладов (27-31 августа 2002 г.). – Переславль: УГП, 2002. – С. 131–132.

307. Белишев, Д.В. Итерационный алгоритм второго порядка оптимизации дискретных управляемых систем [Текст] / Д.В. Белишев // Обобщенные решения в задачах управления: труды междунар. симпозиума: тезисы докладов (27-31 августа 2002 г.). – Переславль: УГП, 2002. – С. 225–227.

308. Ермаков, Д.Е. Анализ технологии передачи информации в медицинских информационных системах [Текст] / Д.Е. Ермаков [и др.] // Современные информационные технологии в диагностических исследованиях: тезисы Междунар. конф. (15 марта 2002 г.). – Днепропетровск, 2002. – С. 145–147.

309. Матвеев, Г.Н. Перспективная экономическая подсистема корпоративной медицинской информационной системы [Текст] / Г.Н. Матвеев [и др.] // Интеллектуальное обеспечение охраны здоровья населения: тезисы Междунар. форума (Кемер, октябрь 2002 г.). – Кемер, 2002. – С. 25–34.

310. Хаткевич, М.И. Объектно-реляционный дуализм в больших информационных системах [Текст] / М.И. Хаткевич // Программные продукты и системы. – 2002. – № 3. – С. 22–25.

311. Виноградов, А.Н. Динамические интеллектуальные системы. Ч. I. Представление знаний и основные алгоритмы [Текст] / А.Н. Виноградов, Л.Ю. Жилякова, Г.С. Осипов // Известия РАН. Теория и системы управления. – М.: Наука, 2002. – № 6. – С. 78–96.

312. Виноградов, А.Н. Динамические интеллектуальные системы. Ч. II. Моделирование целенаправленного поведения [Текст] / А.Н. Виноградов, Л.Ю. Жилякова, Г.С. Осипов // Известия РАН. Теория и системы управления. – М.: Наука, 2003. – № 1. – С. 67–75.

313. Осипов, Г.С. Интеллектуальные динамические системы и целенаправленное поведение [Текст] / Г.С. Осипов // Искусственный интеллект. – К.: Наука і освіта, 2002. – № 2. – С. 221–235.

314. Осипов, Г.С. Технология построения интеллектуальных динамических систем для аерокосмических комплексов различного назначения [Текст] / Г.С. Осипов, В.М. Хачумов, А.Н. Виноградов // Системный анализ и управление аэрокосмическими комплексами: тезисы докладов VII-й Междунар. конф. (Евпатория, 1-7 июля 2002г.). – Евпатория, 2002. – С. 11–13.

315. Хачумов, В.М. Разрядно-параллельные структуры для обработки и анализа изображений [Текст] / В.М. Хачумов // Распознавание образов и анализ изображений, новые информационные технологии: 6-я Междунар. конф. РОАИ-6-2002 (Москва, 21-26 октября 2002 г.). – М., 2002. – Т. 2. – С. 597–601.

316. Bashkin, V.A. Resource bisimulation in nested Petri nets [Текст] / V.A. Bashkin, I.A. Lomazova. // In H.-D. Burkhard, L. Czaja, G. Lindemann, A. Skowron, P.Starke (eds.), Concurrency Specification and Programming (CS&P'2002), Proceedings, Informatik-Bericht Nr.161, Humboldt University of Berlin. – 2002. – Vol. 1. – P. 39-52.

317. Абрамов, С.М. Разработка и опыт експлуатации суперкомпьютеров семейства "СКИФ" [Текст] / С.М. Абрамов, В.В. Анищенко, Н.Н. Парамонов, О.П. Чиж // Информационные системы и технологии: I-я междунар. конф. (IST'2002) (Минск, 5-8 ноября 2002 г.). – Минск, 2002. – С. 115–117.

318. Abramov, S. The Universal Resolving Algorithm and its Correctness: Inverse Computation in a Functioanl Language [Текст] / S. Abramov, R. Gluck // Elsevier, Science of Computer Programming 43. – 2002. – P. 193–229.

319. Абрамов, С.М. О построении высокоскоростной оптической магистрали городской компьютерной сети с учетом особенностей електропитания в районных центрах России [Текст] / С.М. Абрамов, В.П. Котельников, А.Ю. Пономарев, Ю.В. Шевчук // Научный сервис в сети Интернет: труды Всеросс. науч. конф. (Москва, 23-28 сентября 2002 г.). – М.: МГУ, 2002. – С. 244–247.

320. Бахвалов, Н.С. Численные методы в задачах и упражнениях [Текст] / Н.С. Бахвалов, А.В. Лапин, Е.В. Чижонков. – М.: Высшая школа, 2000. – 192 с.

321. Tirtishnikov, E.E. Mosaic ranks for weakly semiseparable matrices. [Текст] / E. E. Tirtishnikov // Notes on Numerical Fluid Mechanics. – 2000. – Vol. 73. – P. 36–41.

322. Винокуров, С.Ф. Полиномиальные разложения булевых функций по образам неоднородных операторов [Текст] / С.Ф. Винокуров, Н.А. Перязев // Кибернетика и системный анализ. – 2000. – № 3. – С. 40–55.

323. Strongin, R.G. Global Optimization with Non-Convex Constraints. Sequential and Parallel Algorithms Kluwer Academic Publishers [Текст] /

R.G. Strongin, Ya.D. Sergeyev. – Dordrecht: The Netherlands, 2000. – 728 p.

324. Gergel, V.P. Sequential and parallel global optimization algorithms using derivatives [Текст] / V.P. Gergel, Ya.D Sergeyev // Computers & Mathematics with Applications. – 1999. – Vol. 37 (4/5).– P. 163–180.

325. Strongin, R.G. Parallel Characteristical Algorithms for Solving Problems of Global Optimization [Текст] / R.G. Strongin, Ya.D. Sergeyev, V.A. Grishagin // Journal of Global Optimization, 10. Kluwer Academic Publishers. Printed in the Netherlands. – 1997.– P. 185–206.

326. Konovalov, A. Virtual Shared Files: Towards User-Friendly Inter-Process Communications [Текст] / A. Konovalov, V. Samofalov, S. Scharf // Parallel computing technologies: 5th international conference; PaCT-99, St. Petersburg, Russia, September 6-10, 1999, Victor Malyshkin (ed.), Lecture Notes in Computer Science. – 1999. – Vol. 1662. – P. 223–228.

327. Листровой, С.В. О возможностях построения интеллектуальных вычислительных систем [Текст] / С.В. Листровой, А.И. Тимочко, А.Ю. Гуль // Радіоелектронні і комп'ютерні системи. – Харків: „ХАІ", 2003. – № 3.– С. 155–163.

328. Жихарев, В.Я. Методы моделирования и дискретной оптимизации вычислительных систем реального времени [Текст] / С.В. Листровой, В.С. Харченко; под ред. В.Я. Жихарева. – Харьков; Житомир: ЖГУ, 2004. – 494 с.

329. Мирошник, М.А. Разработка методов повышения отказоустойчивости и надежности функционирования компонентов телекоммуникационных систем и сетей [Текст] / М.А. Мирошник, В.Г. Котух // Радиотехника: Всеукр. межвед. науч.-техн. сб. – 2011. – Вып. 164. – С. 190–197.

330. Мирошник, М.А. Методы тестового диагностирования телекоммуникационных систем на базе одномерных однородных клеточных сетей [Текст] / М.А. Мирошник, Я.Ю. Королева // Інформаційно-керуючі системи на залізничному транспорті. – 2011. – № 2. – С. 68–72.

331. Мирошник, М.А. Тестопригодное проектирование телекоммуникационных систем на основе двумерных однородных сетей [Текст] / М.А. Мирошник, Я.Ю. Королева, С.В. Панченко // Информационно-управляющие системы на железнодорожном транспорте. – 2011. – №3. – С. 46–53.

332. Мирошник, М.А. Развитие современных направлений цифровых телекоммуникационных систем и сетей [Текст] / В.Г. Котух, М.А. Мирошник, С.Н. Селевко // Радиотехника: Всеукр. межвед. науч.-техн. сб. – 2011. – Вып. 165. – С. 254–258.

333. Мирошник, М.А. Интеллектуальные системы обработки данных в телекоммуникационных сетях [Текст] / М.А. Мирошник, Я.Ю. Королева // Інформаційно – керуючі системи на залізничному транспорті. Перспективные компьютерные управляющие и телекоммуникационные системы: тези доповідей XXIV-ї міжнар. конф. (Алушта, 23-27 вересня

2011 р.). – Харків, 2011. – № 4 (додаток) – С. 138.

334. Мирошник, М.А. Концептуальная модель диагностической инфраструктуры с интеллектуальными свойствами для телекоммуникационных систем [Текст] / М.А. Мирошник, Г.И. Загарий, Л.В. Дербунович // Інформаційно – керуючі системи на залізничному транспорті. Перспективные компьютерные управляющие и телекоммуникационные системы: тези доповіді XXIV-ї міжнар. конф. (Алушта, 23-27 вересня 2011 р.). – Харків, 2011. – № 5 (додаток) – С. 146.

335. Мирошник, М.А. Методы повышения отказоустойчивости телекоммуникационных систем [Текст] / В.Г. Котух, М.А. Мирошник, С.Н. Селевко // Радиотехника: Всеукр. межвед. науч.-техн. сб. – 2011. – Вып. 166. – С. 259–268.

336. Мирошник, М.А. Подход к проектированию компьютерных систем с интеллектуальной диагностической инфраструктурой [Текст] / С.Г. Карпенко, М.А. Ковалева, С.В. Панченко// Інформаційно – керуючі системи на залізничному транспорті. – 2011. – №6. – С. 51-59.

337. Мирошник, М.А. Отказоустойчивость распределенных телекоммуникационных систем. [Текст] / М.А. Мирошник, В.Г. Котух, С.Н. Селевко // Радиотехника: Всеукр. межвед. науч.-техн. сб. – 2011. – Вып. 168. – С. 51–55.

338. Мирошник, М.А. Отказоустойчивые вычислительные системы с реконфигурируемой структурой [Текст] / М.А. Мирошник, Г.И. Загарий // Методи та засоби кодування, захисту й ущільнення інформації: тези доповідей III-ї міжнар. наук.-практ. конф. (Вінниця, 29-31 травня 2012 р.). – Вінниця, 2012. – С. 24.

339. Пристрій для функціонального діагностування пристрою регулювання росту монокристалів [Текст]: пат. 98395 (51) Україна: МПК G01F 11/28 (2006.01), G01R 35/00 / М.А. Бережна, Л.В. Дербунович, Я.Ю. Корольова, В.С. Суздаль; заявник і патентовласник Інститут сцинтиляційних матеріалів НАН України. – №200808602; заявл. 01.07.10; опубл. 11.05.2012, Бюл. №1. – 12 с.

340. Мирошник, М.А. Решение задач диспетчеризации в распределенных телекоммуникационных системах [Текст] / М.А. Мирошник, В.Г. Котух, С.Н. Селевко // Радиотехника: Всеукр. межвед. науч.-техн. сб. – 2011. – Вып. 169. – С. 139–152.

341. Мирошник, М.А. Синтез детерминированных проверяющих тестов для телекоммуникационных систем на одномерных сетях клеточных автоматов [Текст] / М.А. Мирошник, Я.Ю. Королева // Технология приборостроения. – 2012. – № 1. – С. 30–34.

342. Miroshnik, M. Design of a Built-in Diagnostic Infrastructure for Fault-Tolerant Telecommunication Systems [Текст] / M. Miroshnik, N. Miroshnik, S. Panchenko // Інформаційно-керуючі системи на залізничному транспорті. Перспективные компьютерные управляющие и телекоммуникационные системы: тези доповідей XXV-ї міжнар. конф.–

2012. – №4 (додаток). – С. 59.

343. Мирошник, М.А. Исследование методов диагностирования сложных систем [Текст] / М.А. Мирошник, Ю.Н. Салфетникова // Системи обробки інформації. Інформаційні проблеми акустичних, радіоелектронних та телекомунікаційних систем. – 2012. – Вип. 6 (104). – С. 70-75.

344. Мирошник, М.А. Синтез проверяющих тестов для телекоммуникационных сетей на основе циклических отличительных последовательностей [Текст] / М.А. Мирошник, Я.Ю. Королева // Системи обробки інформації. Інформаційні проблеми акустичних, радіоелектронних та телекомунікаційних систем. – 2012. – Вип. 6 (104). – С. 108-113.

345. Мирошник, М.А. Проектирование диагностической инфраструктуры вычислительных систем и устройств на ПЛИС [Текст]: монография / М.А. Мирошник. – Харьков: ХУПС, 2012. – 188 с.

346. Листровой, С.В. Методы и модели планирования ресурсов в GRID-системах [Текст]: монография / С.В. Листровой, С.В. Минухин, В.С. Пономаренко, С.В. Знахур. – Харьков: ИНЖЭК, 2008. – 408 с.

347. Лістровий, С.В. Теорія графів у задачах розподілу ресурсів. Кн. 1 Алгоритми та методи обчислень [Текст]: підручник / Ф.О. Демченко, С.В. Лістровий, М.І. Луханін, Р.В. Семчук. – Харьков: УкрДАЗТ, 2008. – 119 с.

348. Лістровий, С.В. Теорія графів у задачах розподілу ресурсів. Кн. 2: Диферинціально-ігровий підхід до моделювання систем [Текст]: підручник / С.В. Лістровий, О.П. Мартинова, Р.В. Семчук. – Харьков: УкрДАЗТ, 2007. – 143 с.

349. Лістровий, С.В. Основи наукових досліджень [Текст]: підручник / С.В. Лістровий. – Харьков: УкрДАЗТ, 2012. – 338 с.

350. Листровой, С.В. Метод решения задачи о минимальном покрытии как средство планирования в GRID. [Текст] / С.В. Листровой, В.С. Понаморенко // Проблемы управления: РАН. – 2008. – № 3. – С. 78-84.

351. Листровой, С.В. Метод решения задач о минимальном вершинном покрытии и задачи о наименьшем покрытии [Текст] / С.В. Листровой, С.В. Минухин // Электронное моделирование. – 2012. – Т. 34, № 1. – С. 29–43.

352. Листровой, С.В. Подход к формированию оптимальных проектных структур на основе рангового метода решения нелинейных булевых уравнений. [Текст] / С.В. Листровой, С.В. Минухин // Проблемы управления и информатики. – 2011. – №5. – С. 110-121.

353. Listrovoy, S.V. General Approach to Solving Optimization Problems in Distributed Cjmputing Systems and Theory of Intelligence Systems Construction [Текст] / S.V. Listrovoy, S.V. Minykhin // Journal of Automation and Information Sciences. – 2010. – Vol. 42, № 3. – P. 30-45.

354. Листровой, С.В. Об использовании гарантированных прогнозов в методах решения задач булевого программирования на основе рангового

подхода [Текст] / С.В. Листровой, О.Н. Симашкевич // Электронное моделирование. – 2003. – Т. 25, № 4.– С. 89–103.

355. Листровой, С.В. Параллельный алгоритм определения путей с максимальной пропускной способностью [Текст] / С.В. Листровой, В.Н. Хрин // Кибернетика и системный анализ. – 1998. – № 2. – С. 125–134.

356. Listrovoy, S.V. On the class of NP-complete problems and approach Baptism of discrete optimization and graph theory [Текст] / S.V. Listrovoy // LAP LAMBERT Academic Publishing GmbH & Co. KG. – 2014. – 100 p.

357. Комутаційний пристрій – оптоелектронний аналог електромагнітного реле [Текст]: пат. 32964 Україна: МПК (2006) H03K 17/60. – № 200808105; заявл. 14.01.08; опубл. 10.06.2008; Бюл. № 11. – 12 с.

358. Пристрій регулювання росту монокристалів та пристрій діагностування для нього [Текст]: пат. 86105 Україна: МПК7 C30 B 15/20, G06 F 11/28. – №200808602; заявл. 26.04.07; опубл. 25.03.2009, Бюл. № 6. – 14 с.

359. Пристрій для діагностування пристрою регулювання росту монокристалів [Текст]: пат. 89312 Україна: МПК8 G06 F 11/28, C30 B 15/20. №200808602; заявл. 01.07.08; опубл. 11.01.2010, Бюл. № 1. – 8 с.

360. Пристрій для функціонального діагностування пристрою регулювання росту монокристалів [Текст]: пат. 98395 (51) Україна: МПК G01F 11/28 (2006.01), G01R 35/00. №200808602; заявл. 20.12.10; опубл. 10.05.2012, Бюл. №9. – 12 с.

361. Пристрій для рішення задач на графах [Текст]: пат. 69487 Україна: МПК (2012.01), G06F 15/00. – №u201113667; заявл. 21.11.11; опубл. 25.04.2012, Бюл. №8. – 12 с.

362. Miroschnik, M.A. Application of software complex for query processing in the database management system with a view of dispatching problem solving in Grid systems [Текст] / M.A. Miroschnik, V.G. Kotukh, S.N. Selevko // Telecommunications and radio engineering. – 2013. – Vol. 27, № 10. – P. 875-891.

363. Listrovoy, S.V. General Approach to Solving Optimization Problems in Distributed Cjmputing Systems and Theory of Intelligence Systems Construction [Текст] / S.V. Listrovoy, S.V. Minykhin // Journal of Automation and Information Science. – 2010. – Vol. 42, N. 3. – P. 30-45.

364. Miroschnik, M.A. The comparative analysis of a multiprobe microwave multimeters with involvement of processing by the Kalman filtering and the least-squares methods with regard for Re-reflection of probes [Текст] / M.A. Miroschnik, O.B. Zaichenko, I.I. Klyuchnik, R.I. Tzekhmistro // Telecommunications and radio engineering. – 2015. – Vol. 74, № 74 (1). – P. 79-86.

365. Listrovoy, S.V. Jn Cjrrelation of P and NP-Classes [Текст] / S.V. Listrovoy // I.J. Modern Education and Computer Science. – 2012. – № 3. – P. 21-27.

366. Listrovoy, S.V. Znakhur Investigation of the Scheduler for

Heterogeneous Distributed Computing Systems based on Minimal Cover Method [Текст] / S.V. Listrovoy, S.V. Minykhin. // Internation Journal of Computer Aplication (0975-8887). – August 2012. – Vol. 51, No. 19. – P. 123-127.

367. Листровой, С.В. Анализ сетевого трафика стека протоколов TCP/IP на основе динамической модели [Текст] / С.В. Листровой, С.В. Моцний // Інформаційно-керуючі системи на залізничному транспорті. – 2014. – № 5(108). – С. 10-14.

368. Лістровий, С.В. Метод перечисления максимальных независимых множеств в произвольных неориентированных графах. НАНУ [Текст] / С.В. Лістровий // Электронное моделирование. – 2003. – Т. 36, №1. – С. 3-14.

369. Listrovoy, S.V. Algoritm of Sub Exponential Complexity for the SAT [Текст] / S.V. Listrovoy V.M. Butenko // International Jornal of Computer and Information Technology (ISSN: 2279-0764) – September 2013. – Vol. 2. – Issue 05. – P. 211-215.