НАЦІОНАЛЬНА АКАДЕМІЯ НАУК УКРАЇНИ
ІНСТИТУТ КІБЕРНЕТИКИ ім. В. М. ГЛУШКОВА
Вищий навчальний заклад Укоопспілки
«ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ»

Г. П. Донець
Л. М. Колєчкіна

ЕКСТРЕМАЛЬНІ ЗАДАЧІ НА КОМБІНАТОРНИХ КОНФІГУРАЦІЯХ

Монографія

ПОЛТАВА
РВВ ПУЕТ
2011

УДК 519.85
ББК 22.18
Д67

Рекомендовано до видання, розміщення в електронній бібліотеці та використання в навчальному процесі вченою радою університету, протокол № 9 від 16 листопада 2010 р.

Автори:
Г. П. Донець, Л. М. Колєчкіна

Рецензенти: ***Ф. І. Андон,*** д.ф.-м.н., академік НАН України, завідувач відділу економічної кібернетики Інституту кібернетики ім. В. М. Глушкова; ***С. В. Яковлев,*** д.ф.-м.н., професор, завідувач кафедри вищої математики та прикладної інформатики Харківського інституту управління, заслужений діяч науки і техніки України.

Донець Г. П.
Д67 Екстремальні задачі на комбінаторних конфігураціях : монографія / Г. П. Донець, Л. М. Колєчкіна. – Полтава : РВВ ПУЕТ, 2011. – 309 с.

ISBN 978-966-184-133-7

Монографія присвячена розробці та обґрунтуванню математичних моделей і обчислювальних методів розв'язання екстремальних задач дискретної оптимізації на комбінаторних конфігураціях. Особлива увага приділена дослідженню структурних властивостей допустимої області, побудові графів многогранників комбінаторних конфігурацій, що є областями допустимих розв'язків. Побудовано та обґрунтовано нові підходи й ефективні методи розв'язання екстремальних задач на комбінаторних конфігураціях при умові додаткових обмежень, проведено аналіз обчислювального експерименту.

Монографія розрахована на широке коло читачів-математиків, фахівців з математичного моделювання, кібернетики, теорії та практики векторної та дискретної оптимізації, а також студентів і аспірантів відповідних спеціальностей.

УДК 519.85
ББК 22.18

ISBN 978-966-184-133-7

ЗМІСТ

ВСТУП

Останнім часом з'явилася велика кількість праць, присвячених методам розв'язування екстремальних задач на комбінаторних конфігураціях [1–5, 11, 13, 53, 55–60, 86, 88–90, 94–99, 110, 112, 114–118, 120–127, 130, 137, 145–147, 171, 173, 176, 177, 180, 188, 193–196, 200, 202–205, 210–220, 223, 225, 226–228, 272–277, 283], в яких пропонуються нові підходи та удосконалюються існуючі методи, досліджуються їх ефективність. Даний факт пов'язаний з тим, що екстремальні задачі дискретної оптимізації є моделями різних прикладних задач проектування, планування, розміщення, класифікації і управління. Слід зазначити, що в науковій літературі широко вживається термін «задача комбінаторної оптимізації» (ЗКО), під якою розуміють проблему пошуку екстремумів заданої цільової функції на комбінаторному просторі. Під комбінаторним простором тут розуміється сукупність комбінаторних об'єктів певного типу, утворених із елементів заданої скінченної множини. Водночас, в більшості робіт з даного напряму, поняття «комбінаторний об'єкт» не формалізується, а, як правило, найчастіше називаються такі множини як сполучення, перестановки та розміщення. Але для більш строгої формалізації поняття комбінаторного об'єкту може бути здійснене на основі поняття комбінаторної конфігурації.

Отже в подальшому слід розуміти, що комбінаторні конфігурації та комбінаторні множини вживаються в одному й тому ж розумінні, в залежності від специфіки та властивостей задачі, хоча поняття комбінаторної конфігурації є більш об'ємним з точки зору комбінаторики.

Прикладами екстремальних задач є задача комівояжера, транспортна задача, задача розміщення об'єктів, трасування друкованих плат, що виникають при проектуванні обчислювальної апаратури, в автоматизованих системах управління тощо. Моделями таких задач є велике число прикладних задач, що виникають на виробництві [31, 33, 38–40, 52, 53, 92–94, 129, 146, 150–152, 160–166, 239, 268].

Необхідність розв'язання саме цих задач стимулювало розвиток теорії дискретної, зокрема комбінаторної оптимізації. Комбінаторна оптимізація – область математики, предметом якої є

дослідження і розв'язання екстремальних задач на скінченних множинах комбінаторного характеру.

Основна увага в комбінаторній оптимізації приділяється визначенню обчислювальної складності цих задач, розробці методів і алгоритмів їх розв'язання на основі властивостей комбінаторних множин.

Існує безліч різноманітних методів розв'язування екстремальних задач, як точних так і наближених. Систематичне вивчення властивостей комбінаторних множин та їх дослідження описані в багатьох роботах [9, 14, 18–20, 36, 43–53, 58–60, 95, 136, 141–148, 166–169, 238–246, 253–256]. Підвищений інтерес до таких задач обумовлений дослідженнями останніх років в області комп'ютерних технологій при створенні сучасних алгоритмів і програм для розв'язування оптимізаційних задач. Слід зазначити, що задачі комбінаторної оптимізації на комбінаторних множинах невід'ємно пов'язані з комбінаторними многогранниками, що є опуклими оболонками таких множин, і їх властивостями. Властивості многогранника дають можливість звести розв'язання задач на дискретних комбінаторних множинах до їх розв'язання на неперервній допустимій множині. Тому виникає можливість застосовувати класичні методи неперервної оптимізації до розв'язання комбінаторних задач, що не завжди виправдано.

В загальному випадку ***задача комбінаторної оптимізації*** – це задача про максимізацію або мінімізацію функції при заданих обмеженнях та при умові, що на деякі, або навіть на всі, змінні накладена вимога цілочисельності. ЗКО виникає у величезній кількості практичних задач, одначе, багато важливих з теоретичної та прикладної точок зору класи ЗКО є *NP*-повними. Це означає, що для них невідомо та, швидше всього (при виконанні відомої гіпотези $P \neq NP$) не існує поліноміальних алгоритмів розв'язання. В силу вказаних причин гостро актуальною є проблема зменшення часу роботи алгоритмів розв'язання важкорозв'язних ЗКО, зокрема, проблема скорочення переліку можливих допустимих розв'язків в низці методів (гілок та границь, гілок та відсікань, динамічного програмування тощо). Одним з найпопулярніших та плідних підходів до скорочення переліку є поліедральний метод [284–286], якому присвячено безліч робіт. Цей підхід ґрунтується на використанні інформації про структуру многогранника (поліедра), який є опуклою оболонкою всіх

допустимих розв'язків ЗКО. Важливість дослідження даного многогранника підкреслює, наприклад, той факт, що його фасетам (граням максимальної розмірності) відповідають найсильніші правильні відтинання. Структура многогранників допустимості ЗКО досліджувалась в різних роботах [87, 269]. Зокрема вивчались многогранники багатоіндексних транспортних задач [268, 272, 273] та багато інших. Геометричні аспекти теорії розрізів та метрик на поліедрах присвячена монографія M. M. Deza, M. Laurent [315]. В ній різноманітні результати, отримані для поліедрів у незалежних галузях математики, зв'язуються воєдино за допомогою теорії розрізів та метрик, зокрема, за допомогою понять розрізного конусу та розрізного многогранника. В. Н. Шевченко [269] сформулював гіпотезу, яка оголошує, що клас задач цілочисельного лінійного програмування, в яких мінори матриць обмежень обмежені константою d, є розв'язним за поліноміальний час. Добре відомо, що ця гіпотеза справедлива для d = 1. Частково гіпотеза підтверджена С. І. Веселовим та А. Ю. Чирковим [24, 327] для випадку d = 2. Як з теоретичної, так і з практичної точок зору видається дуже важливим побудова алгоритмів знаходження двоїстого опису поліедра (за його фасетами необхідно знайти вершини та конус рецесивних напрямків) і навпаки. Зазначимо, що на сьогодення питання про існування поліноміальних (від входу та виходу) алгоритмів, що розв'язують цю задачу, відкрите [297], та існують алгоритми, статус складності яких поки що невідомий, але які добре себе зарекомендували на практиці. Одним з них є відомий алгоритм Фур'є-Моцкіна [316]. Запропоновані різні модифікації та ефективні програмні реалізації даного алгоритму [107, 271, 305], а також його модифікації для побудови тріангуляцій [272]. Незважаючи на це, можливості для прискорення цих алгоритмів, напевне, не вичерпані. Близьким напрямком досліджень є друга фундаментальна задача, важлива і з теоретичної і з практичної точок зору, – вивчення структури граней поліедра (див., наприклад, 331) та його тріангуляції [272].

Видається важливим дослідження зв'язку між різними класами тріангуляцій точкових конфігурацій (регулярних, слабко регулярних, розгортуваних, симпліціально політопіальних тощо), зокрема, вивчення структури частково упорядкованої множини, побудованої на сімействі розглядуваних тріангуляційних класів відносно відношення включення. В останній час активно роз-

виваються алгоритми рішення опуклих поліедральних задач допустимості [289, 300]. Одним із цікавих обчислювальних підходів є зведення задачі допустимості для поліедра до задачі проекції на нормальний конус, породжений двоїстою системою нерівностей [174], що достатньо близьке до задачі проекції на двоїстий політоп.

На сьогоднішний день в галузі дослідження різних класів комбінаторних моделей, розробки нових методів їх розв'язання велика увага приділяється методам, що ґрунтуються на використанні структурних властивостей комбінаторних множин. Вивчення властивостей комбінаторних множин тісно пов'язане з теорією многогранників та графів. Використання інформації про структуру опуклої оболонки допустимих розв'язків – многогранників, яка є основою для багатьох методів, – один із самих успішних на сьогоднішній день підходів до розв'язання задач комбінаторної оптимізації. Але при розв'язанні таких задач виникають проблеми, пов'язані зі складністю математичних моделей, великим обсягом інформації та ін.

Прикладні задачі, що моделюються екстремальними дискретними задачами, досить часто мають велику вимірність, тому вони є досить складними з обчислювальної точки зору. Головна задача полягає в тому, щоб визначити таке значення аргументу, що належить деякій комбінаторній конфігурації, для якого цільова функція набуває глобального оптимуму. Тобто, необхідно розробити найбільш ефективний алгоритм, що ґрунтується на специфічних властивостях комбінаторних конфігурацій.

Для екстремальних задач комбінаторної оптимізації розроблялися і розробляються поліноміальні алгоритми знаходження оптимального розв'язку, що ґрунтуються на властивостях структури вхідних даних, але таких робіт, порівняно з методами, що ґрунтуються на частковому переборі варіантів, небагато. Одним із підходів до розв'язання такої проблеми є проведення дослідження і аналізу властивостей комбінаторних конфігурацій, на яких визначається цільова функція, що відображають комбінаторну природу задач. Аналізу і дослідженню комбінаторних конфігурацій як аргументу цільової функції, установленню зміни значень функції цілі в залежності від упорядкування елементів вибраної комбінаторної конфігурації і від специфіки структури вхідних даних у літературі достатньої уваги не приділяється. Але слід зазначити, що вивчення властивостей

певних задач з метою виявлення їх характерних властивостей та їх використання для розв'язання поставленої задачі, дає можливість побудови нових підходів та розвитку відомих методів.

Отже, однією з важливих проблем в області дискретної оптимізації є виявлення властивостей комбінаторних конфігурацій в екстремальних задачах, використання яких дозволяло б установити закономірність зміни значень цільових функцій в залежності від упорядкування аргументу і від специфіки та структури множин комбінаторних конфігурацій.

На сьогоднішній день в області дослідження різних класів комбінаторних моделей і розробки нових методів їх розв'язання отримані істотні результати. Принципово вагомий внесок у розвиток дискретної, зокрема, комбінаторної оптимізації зробили такі закордонні вчені: М. Гері, Д. Джонсон, Х. Пападимитріу, К. Стайгліц, Р. Стенлі, Ф. Харарі, В. О. Євмелічев, В. М. Сачков, а також вчені з України: В. С. Михалевич, І. В. Сергієнко, Н. З. Шор, О. О. Євмець, В. О. Перепелиця, Ю. Г. Стоян, С. В. Яковлев та багато інших.

Розробка комплексного підходу до аналізу властивостей задач комбінаторної оптимізації охоплює широкий спектр досліджень комбінаторних функцій, комбінаторних многогранників, комбінаторних конфігурацій, як аргументу цільової функції, результати якого дають можливість покращити існуючі методи розв'язання таких задач і розробити нові методи знаходження оптимального розв'язку, що ґрунтуються на властивостях комбінаторних конфігурацій – актуальні проблеми комбінаторної оптимізації. Досить важливим в цьому аспекті є розгляд екстремальних задач на комбінаторних конфігураціях з використанням теорії графів.

Останнім часом графи і пов'язані з ними методи досліджень використовуються практично в усіх розділах сучасної математики, та особливо це стосується дискретної математики. Апарат теорії графів є потужним засобом моделювання задач управління дискретними процесами і системами. Граф, як математичне поняття, виявився простим, доступним та потужним засобом математичного моделювання складних систем.

На сьогодні створена досить розвинена теорія екстремальних задач на графах, яка включає дослідження структури властивостей різних класів задач, розробку алгоритмів їх розв'язання, отримання оцінок трудомісткості розв'язання та інших аспектів.

Дослідженням задач на графах присвячені роботи таких вчених як Ф. Харарі, О. Оре, І. В. Сергієнко, В. О. Євмелічев, О. О. Зиков, В. О. Перепелиця, Р. І. Тишкевич та ін.

Незважаючи на досить великі досягнення в області дискретної оптимізації, в процесі моделювання виникають класи екстремальних задач, для яких низка питань є ще не дослідженою.

Принципові труднощі, які виникають при моделюванні, пов'язані також із різними видами невизначеності. До їх числа відносяться: наявність багатьох критеріїв оцінки якості рішень, інтервальне задання параметрів задачі тощо. В цих умовах класичні методи виявляються не достатньо придатними для розв'язування задач.

Задачі оптимізації декількох функцій виникають при дослідженні багатьох теоретичних і прикладних проблем. В переважній більшості будь-яка прикладна задача оптимального проектування схем, технологічних пристроїв машин, конструкцій, складання сітьових графіків, планування і управління виробничою і комерційною діяльністю вимагає, щоб шуканий розв'язок знаходився з урахуванням багатьох критеріїв. Останні, як правило, суперечливі в тому розумінні, що якість порівнюваних альтернатив неможливо адекватно визначити одним комплексним критерієм, що демонструє деяку згортку початкових критеріїв. Отже, це означає, що апарат класичної однокритеріальної оптимізації є недостатнім для пошуку і досягнення ефективних розв'язків. В зв'язку з цим широко розвитку набуває дослідження і використання ширшої та більш загальної теорії – багатокритеріальної оптимізації.

Дослідження в області багатокритеріальної оптимізації в даний час особливо інтенсивно стимулюються практичними потребами і розвитком комп'ютерних інформаційних технологій, комп'ютерних мереж. Значний вклад у розвиток багатокритеріальної оптимізації в Україні внесли такі вчені як І. В. Сергієнко, В. С. Михалевич, Н. З. Шор, В. П. Шило, В. А. Трубін, Ю. Ю. Червак, В. А. Перепелиця та інші вчені.

Зокрема, в Запорізькому державному університеті під керівництвом В. А. Перепелиці його учнями проведені дослідження, які стосуються математичного моделювання задач в області економіки і техніки з використанням моделей дискретної багатокритеріальної оптимізації. Досить глибоко вивчені властивості комбінаторних оптимізаційних задач з векторним критерієм,

питання їх складності, розв'язності, стійкості, алгоритмічні проблеми їх розв'язання.

Властивості розв'язності задач векторної оптимізації досліджені за допомогою алгоритму лінійної згортки критеріїв, який є найпоширенішим алгоритмом пошуку елементів множини Парето для векторних задач, а також завдяки широкому залученню методів теорії графів для розв'язування таких задач.

Вченими кафедри обчислювальної математики Ужгородського національного університету під керівництвом Ю. Ю. Червака досліджено низку нових моделей оптимального вибору за багатьма критеріями, які порівнюються між собою при оцінці альтернативних розв'язків задач оптимізації таким чином, що встановлюється апріорно часткова упорядкованість для кожних двох критеріїв, коли один з них важливіший, ніж другий, або ж обидва вони однаково важливі.

Цікаві результати одержані при застосуванні ідей лексикографічної оптимізації до розв'язання однокритеріальних задач дискретної оптимізації, зокрема, до задач, в яких частина або всі змінні є цілочисловими. Відомий загальний підхід до розв'язування задач у вигляді методу лексикографічних відтинань.

Але багато задач проектування, планування, розміщення, управління та інші описуються за допомогою різних моделей багатокритеріальної оптимізації, розв'язки яких мають комбінаторні властивості, зокрема, переставні, сполучні та інші, тобто пошук розв'язків здійснюється на деякій комбінаторній конфігурації.

Як відомо, більшість комбінаторних оптимізаційних задач можуть бути зведені до задач цілочисельного програмування, але це не завжди виправдано, оскільки при цьому втрачається можливість врахування комбінаторних властивостей розв'язків задач.

Дана робота є продовженням досліджень в області екстремальних задач дискретної, зокрема комбінаторної оптимізації, а також задач оптимізації за умови багатокритеріальності, що виникають при дослідженні багатьох теоретичних і прикладних проблем.

У монографії розглядається нова і актуальна задача, яка об'єднує в собі проблему багатокритеріальності і комбінаторні властивості розв'язків; розроблено новий метод розв'язування екстремальних задач на комбінаторних конфігураціях, який

називається методом направленого структурування. Основна ідея запропонованого методу перекликається з відомим методом послідовного аналізу варіантів, суть якого описано в оглядовій частині. Але новий метод призначений безпосередньо для комбінаторних задач. Як відомо, у більшості задач на комбінаторних конфігураціях постає необхідність перераховувати велику кількість варіантів, порівнянну з факторіальною величиною. Тому, метод об'єднує засоби комбінаторного аналізу та теорії графів.

В роботі наведено приклади застосування нового методу з застосуванням теорії графів для розв'язування екстремальних комбінаторних задач з лінійною та дробово-лінійною функціями.

РОЗДІЛ 1. КОМБІНАТОРНІ КОНФІГУРАЦІЇ ТА ЇХ ВЛАСТИВОСТІ

В даному розділі розглядаються різні комбінаторні конфігурації, визначення яких може бути сформульованим на основі загального поняття конфігурації, представленого в термінах відображення множини.

1.1. Визначення комбінаторних конфігурацій, різні види, та їх обмеження

Під множиною розуміється сукупність предметів і понять, об'єднаних деякою загальною властивістю. Іноді замість слова «множина» вживають терміни: сукупність, сімейство й т. д. Множина складається з елементів, і запис $x \in X$ означає, що елемент x належить множині X; в іншому випадку пишуть $x \notin X$, і це означає, що елемент x не належить множині X. Якщо кожний елемент $x \in X$ має властивість $x \in Y$, то говорять, що X є підмножина Y і записують $X \subseteq Y$. Множини X та Y рівні, якщо $X \subseteq Y$ і $Y \subseteq X$, що відповідає запису $X = Y$. X є власною підмножиною Y, $X \subset Y$, якщо $X \subseteq Y$ і $X \neq Y$. Будь-яка множина містить як підмножину порожню множину, що позначається символом $\varnothing$.

Представлення множини можна задати або виписуванням її елементів, або шляхом вказівки їх загальних властивостей. Використовуючи другий спосіб опису множин, визначимо для них так звані булеві операції.

Розглянемо деякі з них:

1. Об'єднання

$$X \bigcup Y = \{x : x \in X \text{ або } x \in Y\}$$

тобто $X \bigcup Y$ є множина елементів, які належать принаймні одній з множин X і Y.

2. Перетин

$$X \bigcap Y = \{x : x \in X,\ x \in Y\}.$$

3. Різниця

$$X \setminus Y = \{x : x \in X,\ x \notin Y\}.$$

4. Доповнення множини X відносно Y, $X \subseteq Y$

$$\overline{X} = Y \setminus X.$$

Сукупність всіх підмножин множини X, включаючи порожню множину, називається булеаном X і позначається 2^x. Елементи $X_1, X_2, \ldots, X_r$ булеана 2^x утворюють розбиття множини X, якщо $X_i \neq \varnothing$, $i = 1, 2, \ldots, r$ і

$$X = X_1 \cup X_2 \cup \ldots \cup X_r;\ X_i \cap X_g = \varnothing,\ i \neq g. \qquad (1.1)$$

Множини $X_1, X_2, \ldots, X_r$ називають блоками розбиття. Сукупність всіх упорядкованих пар (x, y) таких, що $x \notin X$, називається декартовим добутком множин X і Y і позначається $X \times Y$ тобто

$$X \times Y = \{(x, y) : x \in X, y \in Y\}.$$

Аналогічно визначається декартовий добуток r множин

$$X_1 \times X_2 \times \ldots X_r = \{(x_1, x_2, \ldots, x_r)\, x_1 \in X_1, x_2 \in X_2, \ldots, x_r \in X_r\},$$

а при збігу множин – декартова степінь $X^{(r)} = X \times X \times \ldots \times X$.

Нехай X – скінченна множина, а $|X|$ – число її елементів. Сформулюємо очевидне аксіоматичне правило, що лежить в основі багатьох комбінаторних обчислень.

Правило суми. Якщо X – скінченна множина і

$$X = X_1 \bigcup X_2 \bigcup \ldots \bigcup X_r,\ X_i \in 2^x,\ i = 1, 2, \ldots, r, \text{ то}$$
$$|X| \leq |X_1| + |X_2| + \ldots + |X_r|,$$

причому рівність досягається тоді, коли $X_1, X_2, \ldots, X_r$ утворять розбиття X.

Приведемо друге просте правило, що використовується в комбінаторному аналізі.

Правило добутку. Для скінченних множин $X_1, X_2, \ldots, X_r$

$$\left|X_1 \times X_2 \times \ldots \times X_r\right| = \left|X_1\right| \cdot \left|X_2\right| \cdot \ldots \cdot \left|X_r\right|.$$

2. Бінарні відповідності й бінарні відношення *t*. Усяка множина пар $R \subseteq X \times Y$ називається бінарною відповідністю на множинах X і Y. Якщо $(x,y) \in R$ то x і y назвемо проекціями (x,y) на X і Y відповідно й будемо записувати в таким способом:

$$x = \pi_1(x,y), \quad y = \pi_2(x,y).$$

Для бінарної відповідності $R \subseteq X \times Y$ можна визначити її проекцію на X і Y:

$$\pi_1(R) = \{x : x = \pi_1(x,y),\ (x,y) \in R\}$$

$$\pi_2(R) = \{y : y = \pi_2(x,y),\ (x,y) \in R\}.$$

Проекції $\pi_1(R), \pi_2(R)$ іноді називаються відповідно областю визначення й областю значення R. Образом елемента $x \in X$ при відповідності R будемо називати множину

$$\delta_1(x;R) = \{y : y \in Y, (x,y) \in R\}.$$

Аналогічно прообразом елемента $y \in Y$ при відповідності R називається множина

$$\delta_1(y;R) = \{x : x \in R,\ (x,y \in R)\}.$$

Бінарна відповідність $\varphi \subseteq X \times Y$ називається функціональною, якщо образ кожного елемента $x \in X$ містить рівно один елемент.

Для множин $X = (x_1, x_2, \ldots, x_n), Y = \{y_1, y_2, \ldots, y_m\}$ бінарній відповідності $R \subseteq X \times Y$ в множині можна поставити матрицю $A = \|a_{ig}\|, i = 1,2,\ldots,n, g = 1,2,\ldots,m.$

І навпаки, будь-якому розбиттю X відповідає відношення еквівалентності, класи якого збігаються з блоками зазначеного

розбиття. Множина всіх класів еквівалентності називається фактор-множиною по даному відношенню еквівалентності.

Бінарне відношення R на множині X називається відношенням часткового порядку, якщо воно рефлексивне, антисиметричне й транзитивне. Множина X називається в цьому випадку частково впорядкованою множиною. Якщо R – відношення часткового порядку і xRx', то звичайно пишуть: $x \le x'$.

Відношення часткового порядку R, що задовольняє умові дихотомії, будемо називати відношенням лінійного порядку або просто відношенням порядку. Запис $x \le x'$ використовуємо для позначення того, що x і x' зв'язано відношенням R лінійного порядку: xRx'.

На множині X можна визначити відношення строгого часткового (лінійного) порядку – $\prec (\prec)$, вважаючи що $x \prec x' (x \prec x')$, якщо $x \le x' (x \le x')$ і $x \ne x'$.

Ясно, що на скінченій множині X завжди можна задати відношення строгого порядку. Кожне таке відношення визначає перестановку елементів цієї множини, і число таких відношень дорівнює $|X|!$.

На булеані 2^x скінченої множини X можна задати відношення часткового порядку, вважаючи $X \le X'$ тоді й тільки тоді, коли $X \subseteq X'$ для всіх X, $X' \in 2^x$. У свою чергу на декартовій степені X^r скінченої множини X із заданим строгим лінійним порядком можна встановити відношення лінійного порядку, при якому

$$(x_1, x_2, \ldots, x_r) \prec (x_1, x_2, \ldots, x_r), (x_1, x_2, \ldots, x_r') \in X^{(r)},$$

якщо для найменшого індексу i із властивістю $x_i \ne x_i'$ має місце $x_i \prec x_i'$. Такий порядок є лексикографічним і використовується для впорядкування слів у словниках.

Відображення. Закон φ, відповідно до якого кожному елементу $x \in X$ ставиться у відповідність єдиний елемент $\varphi(x) \in Y$, називається однозначним відображенням множини X в множину Y або функцією.

При $X = Y$ бінарну відповідність $R \subseteq X \times X$ будемо називати бінарним відношенням на множині X. Прикладом бінарного відношення на множині X є відношення рівності $\Delta_x \{(x,x) : x \in X\}$.

Бінарному відношенню R на скінченній множині X можна поставити у відповідність геометричний об'єкт, що називається орієнтованим графом або діаграмою. Кожному елементу $x \in X$ ставиться у відповідність точка на площині, що називається вершиною. Якщо $(x,x') \in R$, то точки, визначені x і x', з'єднуються стрілкою, що виходить з x і заходить в x', яку називають дугою. Сукупність вершин і дуг, побудованих таким чином, являє собою діаграму $\Gamma(X,R)$ відношення R.

Для бінарного відношення R на множині X домовимося, що якщо $(x,x') \in R$, то цей факт записується таким способом: xRx'. Використовуючи це позначення, сформулюємо ряд властивостей, якими можуть володіти бінарні відношення:

1) рефлексивність: xRx для всіх $x \in X$;
2) антирефлексивність: $R \bigcap \Delta_x = \varnothing$;
3) симетричність: із xRx' випливає $x'Rx$;
4) антисиметричність: з xRx' і $x'Rx$ випливає $x = x'$;
5) транзитивність: з xRx', $x'Rx''$ випливає xRx'';
6) дихотомія: або xRx', або $x'Rx, x, x' \in X$.

Бінарне відношення R на множині X називається відношенням еквівалентності, якщо воно одночасно рефлексивне, симетричне і транзитивне. Якщо R – відношення еквівалентності і xRx', то звичайно пишуть $x \approx x'$. Множина елементів $K(x) = \{x' : x' \approx x\}$ назвемо класом еквівалентності, що містить $x, x' \in X$. Класи еквівалентності утворюють розбиття множини X.

Закон φ, згідно якого кожному елементу $x \in X$ ставиться у відповідність єдиний елемент $\varphi(x) \in Y$, називається однозначним відображенням множини X в множину Y або функцією, що визначена на X і приймає значення в Y. Цьому відображенню відповідає запис $\varphi : X \to Y$. Очевидно, що однозначне відобра-

ження $\varphi: X \to Y$ являє собою функціональну бінарну відповідність $\varphi \subseteq X \times Y$. Многозначне відображення ψ множини X в множину Y визначимо як закон, за яким кожному елементу $x \in X$ ставиться у відповідність множина $\psi(x) \subseteq Y$, причому не виключається, що $\psi(x) = \varnothing$. Ясно, що многозначному відображенню ψ множини X у Y відповідає деяка бінарна відповідність $\psi \subseteq X \times Y$. Надалі будемо використовувати в основному однозначні відображення.

У зв'язку із цим зупинимося на властивостях однозначних відображень, для зручності опускаючи постійне згадування про однозначність.

Найбільш вживані дві форми запису відображень: функціональна й дворядкова.

Функціональний запис: $y = \varphi(x)$ означає, що при відображенні $\varphi: X \to Y$ елемент $x \in X$ переходить в елемент $y \in Y$. Для скінчених множин X і Y часто використовується дворядковий запис. Якщо $X = \{x_1, x_2, \ldots, x_n\}$, $Y = \{y_1, y_2, \ldots, y_m\}$, то дворядковий запис, що відповідає $\varphi: X \to Y$ має вигляд

$$\varphi = \begin{pmatrix} x_1 & x_2 & \ldots & x_n \\ \varphi(x_1) & \varphi(x_2) & \ldots & \varphi(x_n) \end{pmatrix}.$$

Розглянемо 0,1-матрицю $A = \|a_{ig}\|$, $i = 1, 2, \ldots, n, g = 1, 2, \ldots, m$,

де $a_{ig} \begin{cases} 1, & y_{g=\varphi(x_i)} \\ 0, & y_g \neq \varphi(x_1) \end{cases}$.

Матрицю A назвемо матрицею інцидентності відображення φ множини $X = \{x_1, x_2, \ldots, x_n\}$ в множину $Y = \{y_1, y_2, \ldots, y_m\}$. У силу однозначності відображення φ матриця A в кожному рядку і в кожному стовпцю має тільки одну одиницю. Такі 0, 1-матриці називають елементарними.

Два відображення $\varphi_1: X_1 \to Y_1, \varphi_2: X_2 \to Y_2$ будемо вважати рівними, якщо $X_1 = X_2, Y_1 = Y_2$ або $\varphi_1(x) = \varphi_2(x)$ для всіх $x \in X_1$. Відображення $\varphi': X' \to Y$ є звуженням, або обмеженням

на X' відображення $\varphi: X \to Y$ якщо $X' \subseteq X$ і $\varphi'(x) = \varphi(x)$ для кожного $x \in X'$.

Множина $\varphi^{-1}(y) = \{x : \varphi(x) = y, x \in X\}$ називається повним прообразом елемента y при відображенні $\varphi: X \to Y$. Повний прообраз елемента y збігається із прообразом цього елемента при функціональній бінарній відповідності $R \subseteq X \times Y$, що відповідає відображенню φ.

Відображення $\varphi: X \to Y$ сюр'єктивне, якщо для кожного $y \in Y$ існує таке $x \in X$, що $y = \varphi(x)$. У цьому випадку також говорять, що φ є відображення X на Y. Очевидно, що при сюр'єктивному відображенні повний прообраз кожного елемента $y \in X$ не є порожньою множиною.

Відображення $\varphi: X \to Y$ ін'єктивне, якщо для будь-яких $x, x' \in X$ таких, що $x \neq x'$, треба $\varphi(x) \neq \varphi(x')$.

Повний прообраз кожного елемента $y \in Y$ при ін'єктивному відображенні містить не більше одного елемента.

Відображення $\varphi: X \to Y$ бієктивне, якщо воно одночасно сюр'єктивне і ін'єктивне. Такі відображення називають також взаємно однозначними.

Якщо $X = Y$, то бієктивне відображення $\varphi: X \to X$ називається підстановкою. Кожній підстановці кінцевої множини X відповідає квадратна матриця інцидентності порядку $|X|$, що має в кожному рядку й у кожному стовпці по одній і тільки одній одиниці. Такі матриці називаються підстановочні.

Закони композиції. Законом композиції або операцією (бінарною) на множині X будемо називати відображення $f: X \times X \to X$. Якщо $f(x, y) = z$, де $x, y, z \in X$, то будемо записувати цей факт таким чином: $xTy = z$.

Закон композиції T називається асоціативним, якщо для будь-яких $x, y, z \in X$ $(xTy)Tz = xT(yTz)$.

Якщо закон композиції задовольняє умові $xTy = yTx$ для будь-яких $x, y \in X$, то він називається комутативним.

Закон композиції T є дистрибутивним щодо закону $\perp$, якщо для будь-яких $x, y, z \in X$ виконуються співвідношення

$$xT(y \perp z) = (xTy) \perp (xTz)$$
$$(y \perp z)Tx = (yTx) \perp (zTx).$$

Стосовно закону композиції T введемо поняття одиничного елемента і елемента, оберненого до даного.

Елемент $e \in X$ називається одиничним або нейтральним щодо закону композиції T, якщо для будь-якого $x \in X : xe = ex = x$.

Якщо такий елемент існує, то він єдиний. Нехай закон композиції T має одиничний елемент. Тоді елемент x^{-1} називається оберненим або симетричним до елемента $x \in X$ щодо цього закону, якщо

$$xx^{-1} = x^{-1}x = e.$$

Якщо – X скінчена група, то величину $|X|$ будемо називати її порядком. Нехай X і Y – скінченні групи довільних порядків із законами композиції T і $\perp$ відповідно і $\varphi : X \to Y$ – відображення, при якому для будь-яких $x, x' \in X$ має місце рівність

$$\varphi(xx') = \varphi(x) \perp (x').$$

Таке відображення φ називається гомоморфізмом групи X в групу Y. Якщо $-\varphi$ бієктивне відображення, то гомоморфізм називається ізоморфізмом.

Непуста підмножина Y групи X із законом композиції T називається підгрупою, якщо виконані наступні умови:

1) з того, що $y \in Y$, слідує $y^{-1} \in Y$;

2) з умови $y \in Y, y' \in Y$ слідує $y', y \in Y$.

Якщо в підмножині $Y \subset X$ композиція будь-яких двох елементів з Y також належить Y, то говорять, що Y замкнута множина щодо даного закону композиції.

Якщо X – скінчена група, а $x_1, x_2, \ldots, x_{k-1}$ – представники всіх правих суміжних класів по підгрупі Y, не вважаючи самого Y, то можна представити розкладання у вигляді

$$X = Y \bigcup Yx_1 \bigcup \ldots \bigcup Yx_{k-1}.$$

Якщо в булеані 2^x як закон $\perp$ взяти операцію об'єднання $\cup$, у якості T – операцію перетину $\cap$ підмножин множини X, до зазначених операцій додати операцію доповнення, то одержимо булеву алгебру.

Множина цілих чисел X, у якому законами композиції є звичайне додавання й множення, утворить кільце.

Говорять, що числа $a,b \in Z$ порівнювані за модулем m, якщо їх різниця $a-b$ ділиться на m. Цей факт записують так: $a \equiv b(mob\ m)$. Порівняння за модулем є відношення еквівалентності, а відповідні класи множин називаються класами лишків за модулем m. Кожний із класів містить у точності одне із чисел $0,1,\ldots,m-1$, які є представниками класів і утворять повну систему найменших недоданих лишків за модулем m.

Позначимо через K_i – клас, представником якого є число $i, 0 \le i \le m-1$.

На множині класів лишків $K_0, K_1, \ldots, K_{m-1}$ можна ввести операції додавання та множення $+$ і $\cdot$. Ці операції над зазначеними класами будемо визначати таким способом:

$K_i + K_g = K_l$, якщо $i + g = l(mod\ m)$,

$K_i \cdot K_g = K_l$ якщо $i \cdot g \equiv l(mod\ m)$.

У результаті одержуємо кільце класів лишків за модулем m, що будемо позначати через Z_m.

У кільці нейтральні елементи щодо законів композиції $\perp$ й T називаються відповідно нулем і одиницею.

Якщо сукупність відмінних від нуля елементів щодо закону композиції T утворить абелеву групу, то в цьому випадку кільце називається полем.

Поле, що містить скінчене число елементів, називається скінченим. Якщо p – просте число, то сукупність класів лишків за модулем p утворить скінчене поле, що містить p елементів.

На основі вище введених понять розглянемо властивості різних видів комбінаторних конфігурацій.

1.2. Властивості різних видів комбінаторних конфігурацій

При розв'язанні комбінаторних задач часто використовуються такі добре відомі конструкції з елементів скінченої множини, як сполучення, розміщення, перестановки тощо. З точки зору комбінаторики, сполученням об'єму m називається підмножина з m елементів вихідної множини $Y = \{y_1, y_2, \ldots, y_n\}$. Якщо для вибраних m елементів є істотним порядок, то говорять про розміщення m елементів. При $m = n$ розміщення являє собою перестановку елементів множини Y. В результаті одержимо сполучення, розміщення, перестановки з повтореннями.

Уже для цих найпростіших комбінаторних конструкцій виникає необхідність формалізації їх визначення з метою уникнути словесних накопичень і плутанини. З ускладненням конструкцій така необхідність стає актуальнішою. Згадана формалізація може бути здійснена у більшості класів випадків шляхом введення поняття конфігурації.

Більш строго формалізація комбінаторних об'єктів може бути здійснена на основі запропонованого в [206, 207] поняття комбінаторної конфігурації.

В означенні комбінаторної конфігурації важливу роль грає формалізація поняття нерозрізненості об'єктів. В даний час добре відомий підхід до визначення нерозрізненості в комбінаторному аналізі, що використовує поняття класів еквівалентності щодо деякої групи підстановок G, яка діє на вхідній множині X. При такому підході множина нерозрізнених об'єктів, або, як часто говорять, задача перерахування, зводиться до визначення коефіцієнтів деякого поліному, що залежить від циклових класів групи G. Даний метод, що одержав у комбінаторному аналізі назву теорії перерахування Пойа, бере свій початок з робіт І. Редфілда й Г. Пойа, а його подальший розвиток можна знайти в роботі де Брейна. Слід зазначити, що більшу частину теорії перерахування Пойа можна застосувати в розробці підходів до розв'язання задач перерахування і насамперед у теорії графів. Однак у цілому ряді важливих випадків застосування методу Пойа зустрічає технічні складності й стає малоефективним. Це насамперед відноситься до таких відомих комбінаторних схем, як розміщення, сполучення, перестановки з різноманітними властивостями, заповнення комірок предметами з обме-

женнями на ємність комірок і вказівкою критеріїв розпізнавання заповнень і т. п.

Тому, одним із важливих понять в даному розділі є визначення загальної комбінаторної схеми та її використання до розв'язування екстремальних задач. Ця схема як часткові випадки охоплює відомі комбінаторні схеми, у тому числі згадувані вище розміщення, перестановки, сполучення, розбиття і т. п. Вона дозволяє сформулювати методику побудови часткових комбінаторних схем, об'єднаних деякими загальними ознаками.

Для формалізації комбінаторних схем використовується поняття відображення однієї скінченої множини X у іншу Y, шляхом введення поняття конфігурації. За цих умов кожному заповненню m предметів в n комірках ставиться у взаємно однозначну відповідність деяка конфігурація $\varphi : X \to Y$, $|X| = m$, $|Y| = n$, а розрізнення заповнення точно визначається властивостями конфігурації φ. Можливий і інший підхід до формалізації комбінаторних схем розглянутого вигляду, що також використовує поняття відображення. Розглянемо суть цього підходу на прикладі урнової схеми.

Розглянемо модель заповнення m різних предметів в n різних комірках. У цьому випадку кожне заповнення знову визначається відображенням $\varphi : X \to Y$, $|X| = m$, $|Y| = n$, причому на φ не накладено ніяких обмежень. Для побудови моделі, у якій комірки різні, а предмети нерозрізнені(однакові), на множині заповнень, що відповідають вихідній моделі, можна задавати відношення еквівалентності, при якому два заповнення еквівалентні, якщо вони відрізняються тільки нумерацією предметів. Ця еквівалентність індукує відповідне відношення еквівалентності на множині відображень, що відповідають заповненням.

Наступний за формалізацією етап у розв'язанні комбінаторної задачі полягає в побудові твірної функції для перерахування різних об'єктів відповідно до деякої характеристики, що називається іноді вагою.

Типовими характеристиками об'єктів є так звані первинні й вторинні специфікації відповідних їм відображень. Можна сказати, що специфікації визначають, у деякому розумінні, склад елементів, що є образами відображення з урахуванням повто-

рень цих елементів. Нехай кожна із груп G і H може збігатися або з одиничною, або із симетричною групою відповідного ступеня. Використовуючи як характеристики первинні, або вторинні специфікації, у кожному з чотирьох випадків, що при цьому виникають, можна вказати спосіб побудови виробничих функцій для перерахування нееквівалентних відображень при різноманітних обмеженнях на ці специфікації. Дану формалізацію певного класу комбінаторних задач разом з методикою побудови відповідних виробничих функцій називають загальною комбінаторною схемою.

Дамо визначення комбінаторної конфігурації, що є основним поняттям в комбінаторній схемі.

Нехай $X=\{1,2,\ldots,m\}$, $Y=\{y_1,y_2,\ldots,y_n\}$ – скінчена лінійно впорядкована множина, і нехай на Y задано строгий лінійний порядок: $y_1<y_2<\ldots<y_n$.

Будемо говорити, що задано відображення φ множини X в множину Y, якщо кожному елементу $x\in X$ ставиться у відповідність єдиний елемент $y\in Y$. При відображенні φ відповідність між x і y записується рівністю $y=\varphi(x)$, а самому відображенню відповідає запис $\varphi: X\to Y$. Множина X є область визначення відображення, а Y – область його значень.

Визначення 1.1. Відображення $\varphi: X\to Y$, що задовольняє деякому комплексу обмежень Λ, будемо називати конфігурацією.

Отже, під комбінаторною конфігурацією розуміється відображення $\varphi: X\to Y$, яке задовольняється певній системі обмежень Λ.

На основі цього підходу побудовані комбінаторні конфігурації, які відповідають найпростішим комбінаторним об'єктам: розміщенням, перестановкам, комбінаціям, розбиттям та іншим об'єктам, утвореним на основі урнової схеми [206].

Комплекс обмежень Λ, якому задовольняє відображення φ, визначає деякий клас конфігурацій, що відповідають умовам на комбінаторній конструкції в розглянутій задачі.

Розглянемо деякі властивості відображень. Відображення $\varphi: X\to Y$ називається сюр'єктивним, якщо будь-який елемент $y\in Y$ має хоча б один прообраз при цьому відображенні.

Іншими словами, для кожного $y \in Y$ існує $x \in X$ такий, що $y = \varphi(x)$. У цьому випадку іноді говорять, що φ відображає X на Y. Якщо φ – сюр'єктивне відображення, то $\varphi(X) = Y$. Для скінчених множин X і Y сюр'єктивність відображення $\varphi : X \to Y$ означає, що $|X| \geq |Y|$.

Відображення $\varphi : X \to Y$ називається ін'єктивним, якщо для будь-якого $y \in Y$ його повний прообраз $\varphi^{-1}(y)$ містить не більше одного елемента. Іншими словами, відображення $\varphi : X \to Y$ ін'єктивне, якщо для будь-яких $x \neq x'$, $x, x' \in X$, $\varphi(x) \neq \varphi(x')$. Якщо X й Y скінченні, то ін'єктивність відображення $\varphi : X \to Y$ означає, що $|X| \leq |Y|$.

Відображення $\varphi : X \to Y$ називається бієктивним, якщо воно одночасно сюр'єктивне й ін'єктивним. Із сюр'єктивності слідує, що $|\varphi^{-1}(y)| \geq 1$ для кожного $y \in Y$; з ін'єктивності випливає умова $|\varphi^{-1}(y)| \leq 1$ для кожного $y \in Y$. Отже, бієктивність відображення φ означає, що $|\varphi^{-1}(y)| = 1$ для будь-якого $y \in Y$, тобто умова $y = \varphi(x)$ для кожного $y \in Y$ однозначно визначає єдине значення $x \in X$. У цьому випадку говорять, що бієктивне відображення φ встановлює взаємнооднозначну відповідність між множинами X й Y. Коли X й Y скінченні, це означає рівність $|X| = |Y|$.

На основі зазначених властивостей відображень опишемо побудову класів конфігурацій, що відповідають найпростішим комбінаторним конструкціям: розміщенням, сполученням, перестановкам, заповненням комірок елементами; тобто дамо строгі визначення цих конструкцій, якими будемо користуватися надалі. Поняття конфігурацій, пов'язаних з вибором і розташуванням деяких об'єктів відповідно до певних правил.

Нехай $\Lambda = \varnothing$, тобто відсутні будь-які обмеження на відображення $\varphi : X \to Y$. Тоді кожна конфігурація φ визначає комбінаторну конструкцію, названу розміщенням з необмеженими пов-

тореннями об'єму m з n різних елементів або m-перестановкою. Очевидно, що число таких конфігурацій дорівнює

$$\cup(n,m)=n^m.$$

Нехай $\Lambda = I$, де I − множина ін'єктивних відображень, тобто множина таких відображень, для яких заборонено два різні елементи $x_i, x_j \in X$ відображати в один елемент $y_i \in Y$. У цьому випадку конфігурація $\varphi: X \to Y$ називається розміщенням об'єму m з n різних елементів або m-перестановкою.

Число таких конфігурацій дорівнює $A(n,m) = n(n-1)\ldots(n-m+1) = (n)_m$, $m \le n$, де $(n)_m = n!/(n-m)!$ і $A(n,m)=0, m>n$. Числа $A(n,m)$ задовольняють рекурентному співвідношенню

$$A(n,m) = A(n-1,m) + mA(n-1,m-1).$$

При $m=n$ кожній конфігурації відповідає перестановка, і їх число дорівнює $P(n)=n!$.

Множину перших k натуральних чисел позначимо N_k, а $N_0^k = N_k \cup \{0\}$.

Якщо n елементів містять q_i i-го виду, де $i \in N_k$, $q_1+q_2+\ldots+q_k=n$ і елементи одного виду ідентичні, то кожній перестановці цих елементів відповідає конфігурація $\varphi: X' \to Y'$, де $X'=\{1,2,\ldots,n\}$, $Y'=\{y_1,y_2,\ldots,y_k\}$, і y_i використовується як образ q_i раз. Число таких перестановок дорівнює

$$P(q_1,q_2,\ldots,q_k)=\frac{n!}{q_1!\,q_2!,\ldots,q_k!}.$$

Визначимо конфігурацію як Λ однозначне відображення, що задовольняє комплексу умов $\varphi: X \to Y$, де $X=\{1,2,\ldots,m\}$, Y − скінчена множина. Далі розглянемо конфігурації σ, для яких усякі обмеження відсутні, тобто $\Lambda = \varnothing$.

Позначивши M множину строго монотонних функцій, тобто таких, що $\varphi(i)<\varphi(j)$, коли $i<j$. Тоді маємо при $\Lambda = M$ відображення сполучень, оскільки умова $\Lambda = M$ визначає заборону монотонності елементів, а отже відсутність відношення порядку.

Нехай σ – конфігурація, що представляє собою відображення множини $X=\{1,2,\ldots,m\}$ в множину $A=\{a_1,a_2,\ldots,a_n\}$. Конфігурація σ може бути записана у вигляді

$$\sigma=\begin{pmatrix}1 & 2 & \ldots & m\\ \sigma(1) & \sigma(2) & \ldots & \sigma(m)\end{pmatrix}=\begin{pmatrix}1 & 2 & \ldots & m\\ a_{i_1} & a_{i_2} & \ldots & a_{i_m}\end{pmatrix},$$

де $\sigma(j)=a_{i_j}$, $j\in N_m$. Іноді зручно конфігурацію σ записувати у вигляді m-мірного вектора з компонентами з множини $A:\sigma=(\sigma(1),\sigma(2),\ldots,\sigma(m))$ або $\sigma=(a_{i_1},a_{i_2},\ldots,a_{i_m})$.

Дамо визначення двох зв'язаних між собою характеристик конфігурації σ, названих первинною й вторинною специфікаціями.

Визначення 1.2. Вираз $[\sigma]=\left[a_1^{\alpha_1},a_2^{\alpha_2},\ldots,a_n^{\alpha_n}\right]$, $\alpha_1+\alpha_2+\ldots+$ $+\alpha_n=m$ називається первинною специфікацією конфігурації σ, якщо вектор $(a_{i_1},a_{i_2},\ldots,a_{i_m})$ має α_j елементів a_i із множини A, де $j\in N_n$. Число α_j називається a_i – показником первинної специфікації σ.

Якщо множину A піддамо перетворенню, що полягає в розмноженні елемента a_i в α_j в ідентичних екземплярах, де $\alpha_1+\alpha_2+\ldots+\alpha_n=m$, то в результаті одержимо мультимножину A' образів відображення $\sigma: X\to A$. Зазначимо, що мультимножина є об'єднання не обов'язково різних елементів; її можна вважати множиною, в якій кожному елементу поставлено у відповідність додатне ціле число, що називається кратністю. Скінчену множину A' можна записувати в наступному вигляді:

$$A=\{a_1,a_2,\ldots,a_n\},$$

де $a_1, a_2, \ldots, a_n$ – обов'язково різні елементи A'. Потужність множини A' позначимо через $|A'|$, і для виписаної вище множини $|A'| = n$. Якщо A' – скінчена множина, то будемо записувати її в наступному вигляді:

$$A = \left\{ \underbrace{a_1, a_1, \ldots, a_1}_{\alpha_1 \text{ раз}}, \underbrace{a_2, a_2, \ldots, a_2}_{\alpha_2 \text{ раз}}, \ldots, \underbrace{a_n, a_n, \ldots, a_n}_{\alpha_n \text{ раз}} \right\} =$$

$$= \{\alpha_1 \cdot a_1, \alpha_2 \cdot a_2, \ldots, \alpha_n \cdot a_n\}.$$

Тут всі a_i – різні і α_i – кратність елементу a_i. В цьому випадку потужність A' рівна

$$|A'| = \sum_{i=1}^{n} \alpha_i .$$

Склад елементів мультимножини A' однозначно визначається вектором $(\alpha_1 + \alpha_2 + \ldots + \alpha_n)$. Будемо говорити, що мультимножина A' має первинну специфікацію $\left[a_1^{\alpha_1}, a_2^{\alpha_2}, \ldots, a_n^{\alpha_n} \right]$, тому що вектор $\left(a_{i_1}, a_{i_1}, \ldots, a_{i_m} \right)$, що відповідає конфігурації σ, та первинній специфікації σ і A' збігаються.

Визначимо відмінність між первинними специфікаціями відображення σ й мультимножиною його образів. Нехай серед чисел $\alpha_1, \alpha_2, \ldots, \alpha_n$, що визначають первинну специфікацію конфігурації σ є β_0 нулів, β_1 одиниць, β_2 двійок і т. д. Тоді, вираз

$$\left[[\sigma] \right] = \left[\left[0^{\beta_0} 1^{\beta_1} 2^{\beta_2} \ldots m^{\beta_m} \right] \right],$$

$\beta_0 + \beta_1 + \ldots + \beta_m = n, \qquad \beta_1 + 2\beta_2 + \ldots + m\beta_m = m,$ називається вторинною специфікацією конфігурації σ. Число β_j називається j-показником вторинної специфікації. Іноді 0-показник вторинної специфікації σ можна опускати,

записуючи $\left[\left[\sigma\right]\right] = \left[\left[1^{\beta_1} 2^{\beta_2} \dots m^{\beta_m}\right]\right]$, $\beta_0 + \beta_1 + \dots + \beta_m = n$, $\beta_1 + 2\beta_2 + m\beta_2 = m$.

Аналогічним чином можна говорити про те, що мультимножина образів відображення має вторинну специфікацію $\left[\left[0^{\beta_0} 1^{\beta_1} 2^{\beta_2} \dots m^{\beta_m}\right]\right]$, $\beta_0 + \beta_1 + \dots + \beta_m = n$, $\beta_1 + 2\beta_2 + \dots + m\beta_m = m$, якщо число елементів множини A', що зустрічається в A' i раз, дорівнює $\beta_i \left(i = 0,1,\dots,m\right)$. Множину всіх відображень $\sigma : X \to A$ будемо позначати A^x. Ясно, що між A^x і m-й декартовим ступенем $A^{(m)}$ існує бієктивна відповідність.

У дискретній математиці важливу роль грають комбінаторні методи, необхідність у застосуванні яких виникає щораз, коли цікавляться існуванням відповідних алгоритмів та числом способів розташування елементів деякої множини відповідно до заданих правил. Кожне таке розташування визначає комбінаторну конфігурацію, яку можна розглядати як відображення однієї множини в іншу з деякими обмеженнями, обумовленими конкретними вимогами до задачі.

Якщо обмеження носять складний характер, то для відповідних комбінаторних конфігурацій необхідно з'ясувати умови існування й знаходити методи їх конструювання.

Для подальшого викладу матеріалу розглянемо поняття елементу довільної комбінаторної конфігурації як вершини деякого графа.

Як відомо, під комбінаторним об'єктом розуміють підмножину заданої дискретної множини, що задовольняє деяким властивостям.

Якщо розглядати задачі комбінаторної оптимізації, то при накладанні певних умов вони мають дискретно-неперервну структуру. Це обумовило можливість використання як континуальних, так і дискретних моделей зазначених задач. При проектуванні вхідних даних комбінаторної задачі при відображенні *P*-простору вхідної інформації *G* на простір результуючої інформації G^*, виникає етап неперервної оптимізації, обумовлений релаксацією задачі.

Розглянемо задачі, моделями яких є екстремальні задачі оптимізації на перестановках, розміщеннях, сполученнях та ін.

Основна увага при цьому приділяється дослідженню фундаментальних властивостей зазначених множин та розгляд елементів таких множин як вершин деяких графів. Ці властивості дозволяють застосовувати відомі підходи до розв'язання задач комбінаторної оптимізації та розвинути нові. Розглянемо властивості типових видів комбінаторних конфігурацій.

1.2.1. Властивості перестановок

Практика показала, що найчастіше комбінаторними об'єктами є перестановки. Зокрема, перестановкою із n елементів називається така впорядкована множина з n елементів, які різняться між собою порядком їх розміщення. Надалі, якщо не оговорюється протилежне, будемо розглядати комбінаторні множини, породжені дійсними числами.

Розглянемо необхідні поняття, що визначають комбінаторні множини.

Вимірність підпростору $L \subset R^k : dim\, L$. Опуклу оболонку множини M позначимо $conv\, M$.

Нехай задана мультимножина $A = \{a_1, a_2, \ldots, a_q\}$, її основа $S(A) = \{e_1, e_2, \ldots, e_k\}$, тобто набір її різних елементів, де $e_j \in R^1 \ \forall j \in N_k$ і кратність елементів $k(e_j) = \alpha_j$, $j \in N_k$, $\alpha_1 + \alpha_2 + \ldots + \alpha_k = q$. Мультимножина B з основою $S(B)$ називається підмультимножиною мультимножини A з основою $S(A)$, якщо $S(B) \subset S(A)$ і для кожного елемента $a \in S(B)$ виконується нерівність $k_B(a) \le k_A(a)$.

Впорядкованої n-вибіркою з мультимножини A називається набір:

$$a = \left(a_{i_1}, a_{i_2}, \ldots, a_{i_n}\right), \tag{1.3}$$

де $a_{i_j} \in A \ \ \forall i_j \in N_n$, $\forall j \in N_n$, $i_s \ne i_t$, якщо $s \ne t \ \forall s \in N_n$, $\forall t \in N_n$.

Множина перестановок з повтореннями з n дійсних чисел, серед яких k різних, називається загальною множиною перестановок і позначається $P_{nk}(A)$. Це множина упорядкованих n-вибірок вигляду (1.3) з мультимножини A за умови $n = q > k$.

При $n = k = q$ маємо множину перестановок без повторень. Позначимо її P_n. Очевидно, що $P_n(A) = P_{nn}(A)$. У тих випадках, коли конкретно не будемо вказувати вид множини перестановок, будемо записувати ці множини як P_n. Елементами множини перестановок P_n є упорядковані набори $a = (a_1, a_2, \ldots, a_n)$ із n символів. Кожен символ a_i входить у набір тільки один раз і є представником множини $A_i, i \in N_n$.

Елементами множини перестановок з повтореннями P_n є упорядковані набори, що складаються із n символів, причому кожен символ $a_i \in A_i, i \in N_k, k \le n$ входить у набір n_i раз, а $\sum_{i=1}^{k} n_i = n$, тоді розглядається множина перестановок з повтореннями.

Якщо розглядати множину P_{nk}, то зрозуміло, що всяка точка $a = (a_{j_1}, a_{j_2}, \ldots, a_{j_n}) \in P_{nk}$ має таку властивість, що її координати приймають α_1 значень a_1, α_2 значень a_2 і т. д., α_k значень a_k.

Слід зазначити, що опуклою оболонкою загальної множини перестановок є загальний переставний многогранник, властивості якого описано в роботах [87, 245].

Далі розглянемо властивості розміщень та сполучень.

1.2.2. Властивості розміщень та сполучень

Розглянемо множину *k*-розміщень без повторення, тобто A – множина, $q = k$, $A = S(A)$. За такої умови множину всіх упорядкованих *n*-вибірок з мультимножини A вигляду (1.2) називають множиною *n*-розміщень без повторення з k різних дійсних чисел множини A. Позначимо цю множину розміщень $A_k^n(A) = A_q^n$. Якщо $n = k$, то множина A_q^n є множиною $P_n(A)$ перестановок *n*-різних дійсних чисел, що складають A.

Розглянемо множину *q*-розміщень з повтореннями з k різних дійсних чисел з мультимножини A, в якій q елементів.

Розглянемо загальну множину *n*-розміщень.

Нехай $A=\{a_1,a_2,\ldots,a_q\}$ як і раніше – мультимножина з основою $S(G)=(e_1,e_2,\ldots,e_k)$ та первинною специфікацію $[A]=(q_1,q_2,\ldots,q_k)$, де $q_i \le k \forall i \in N_k$. Сукупність усіх упорядкованих *n*-вибірок вигляду (1.3) з мультимножини *A* будемо називати загальною множиною *n*-розміщень і позначити її $A_{qk}^n(A)=A_{qk}^n$. Зазначимо, що оскільки елементи первинної специфікації мультимножини *A* задовольняють умови $q_i \le k$, то в кожному елементі множини A_{qk}^n не більше q_i елементів $e_i \in S(A) \forall i \in k$. Якщо ж $q_i < k \ \forall i \in k$, то, очевидно, що в кожному елементі загальної множини розміщень $A_{qk}^n(A)$ немає однакових чисел e_i, $i \in N_k$ (на відміну від множини $A_{qk}^n(A)$, де є k елементів, що складаються з однакових чисел e_i, $i \in N_k$). Зазначимо також, що при $k=q$, тобто при $q_i=1 \ \ \forall i \in N_k$, множина $A_{qk}^n(A)$ збігається з множиною $A_k^n(A)=A$ розміщень без повторень: $A_{kk}^n(A)=A_k^n(A)$, а при $q_i=n \ \forall i \in N_k$, тобто при $q=nk$, множина $A_{qk}^n(A)$ перетворюється на множину $A_k(A)$ розміщень з повтореннями: $A_{qk}^n(A)=A_k^n(A)$.

Якщо $n=q$, то множина $A_{qk}^n(A)$ є загальною множиною перестановок $P_{nk}(A)$ з елементів мультимножини *A*.

Як відомо [49, 51, 169], опуклою оболонкою точок загальної комбінаторної множини розміщень є загальний многогранник розміщень $M_{qk}^n(A)=conv\,A_{qk}^n(A)$. Загальний многогранник розміщень $M_{qk}^n(A)$ можна представити сукупністю всіх розв'язків такої системи нерівностей:

$$\sum_{j=1}^{|\omega|} a_{q-j+1} \le \sum_{i \in \omega} x_i \le \sum_{j=1}^{|\omega|} a_j, \forall \omega \subset N_n.$$

Визначення 1.3. Вектор *x* є вершиною загального многогранника розміщень $M_{qk}^n(A)$ тоді і тільки тоді, коли він

представляє собою перестановку чисел, $a_1,\ldots,a_s,\ldots,a_{q-r+1},\ldots,a_q$, де $s\in N_n, r\in\{N_n\cup 0\}$, $s+r=n$.

Розглянемо сполучення. Множина всіх *k*-вибірок з множини $S(A)$ вигляду $\left(a_{i_1},a_{i_2},\ldots,a_{i_k}\right)$ називають множиною *k*-сполучень без повторення з n різних дійсних чисел, якщо виконується умова $e_{i_1}<e_{i_2}<\ldots<e_{i_k}$. Позначимо цю множину сполучень $S_n^k(A)$. Сукупність всіх *k*-вибірок вигляду (1.3) з мультимножини A називається загальною множиною сполучень $S_{qn}^k(A)$, якщо виконується умова $a_{i1}\le a_{i2}\le\ldots\le a_{ik}$.

1.2.3. Властивості розбиття

При розв'язанні комбінаторних задач часто використовується такі добре відомі конструкції з елементів скінченної множини, як сполучення, розміщення, перестановки і т. д. Але поряд з добре відомими комбінаторними конфігураціями виділяються складніші конструкції – множини розбиття. Інтерес до таких множин обумовлений різними прикладними задачами, які добре описуються за їх допомогою. У літературі розглядається розбиття множини і розбиття числа. Розглянемо дані поняття більш докладніше.

Сукупність $A_1,A_2,\ldots,A_k$ не порожніх попарно диз'юнктивних підмножин множини *M* називається його розбиттям, якщо множина *M* рівна їх об'єднанню:

$$M=A_1\cup A_2\cup\ldots\cup A_k, A_i\cap A_j=\varnothing, i\ne j.$$

Розбиття $A=\{A_1,A_2,\ldots,A_k\}$ множини *M* називається дробленням розбиття $B=\{B_1,B_2,\ldots,B_l\}$ тієї ж множини, якщо кожна із множин A_i, повністю міститься в одній із множин B_j, тобто для кожного $i\in N_k$ знайдеться єдине $j\in N_l$, таке що $A_i\in B_j$. Будемо говорити, що *B* складніше (крупніше), ніж *A*.

Розглядаються впорядковані і невпорядковані розбиття. Розглянемо впорядковані розбиття множини. Визначимо число розбиттів скінченої множини *S*, де $|S|=n$, на k різних

підмножин $S = S_1 \cup S_2 \cup \ldots \cup S_k$, що попарно не перетинаються, $|S| = n_i$, $i = 1, 2, \ldots, k$ і $\sum_{i=1}^{k} n_i = n$. Послідовність різних множин $S_1, S_2, \ldots, S_k$ розглядається як упорядкована послідовність підмножин. При формуванні упорядкувань $S_1, S_2, \ldots, S_k$ послідовностей на перше місце підмножину S_1 можна вибрати $C_n^{n_1}$ способами, на друге місце підмножину S_2 можна вибрати $C_{n-n_1}^{n_2}$ способами із $n - n_1$ елементів, що залишилися і т. д., на останнє місце множину S_k можна вибрати $C_{n-n_1-n_2-\ldots-n_{k-1}}^{n_k}$ способами із $n - n_1 - n_2 - \ldots - n_{k-1}$ елементів, що залишилися. За правилом прямого добутку отримуємо, що загальне число впорядкованих розбиттів множини *S* на *k* підмножин рівно

$$C_n^{n_1} C_{n-n_1}^{n_2} \ldots C_{n-n_1-\ldots-n_{k-1}}^{n_k} = \frac{n!}{n_1! n_2! \ldots n_k!},$$

що співпадає з числом $P(n_1, n_2, \ldots, n_k)$ перестановок з повтореннями.

Зауваження. Встановимо взаємооднозначну відповідність між впорядкованим розбиттям множини і перестановками з повтореннями. Кожній перестановці з повтореннями можна поставити у відповідність впорядковане розбиття множини номерів елементів $S = \{1, 2, \ldots, n\}$ у перестановці на підмножини $S_1, S_2, \ldots, S_k$ де S_i – множина номерів елементів *i*-го типу в перестановці. Очевидно, що дана відповідність між перестановками з повтореннями і розбиттям є взаємооднозначною.

Загальна кількість елементів невпорядкованого розбиття буде в $m_1! m_2! \ldots m_n!$ раз менше, ніж впорядковане. Отже

$$N(m_1, m_2, \ldots, m_n) = \frac{n!}{(1!)^{m_1} (2!)^{m_2} \ldots (n!)^{m_n} m_1! m_2! \ldots m_n!}.$$

Зазначимо, що якщо виконано впорядковане розбиття числа на *n* підмножини різної довжини (потужності), то вони

співпадають з невпорядкованим розбиттям. В цьому випадку всі $m_i \notin \{0,1\}$.

Формула

$$(x_1 + x_2 + \ldots + x_k)^n = \sum_{n_1+n_2+\ldots+n_{k=n}} \frac{n!}{n_1!n_2!\ldots n_k!} x_1^{n_1} x_2^{n_2} \ldots x_k^{n_k}$$

називається поліноміальною, де додавання виконується по всім розв'язкам рівняння $n_1 + n_2 + \ldots n_k = n$ у цілих невід'ємних числах $n_i \geq 0,\ i = 1, 2, \ldots, k$.

Розбиття часто виникає при формуванні повних прообразів значень функції: якщо взяти деяке відображення $f : M \to R$ і співставити кожному $r \in R$ множину $A_r = \breve{f}(r) = \{i \in M \mid f(i) = r\}$, то набір непорожніх множин A_r матиме всі властивості розбиття. Позначимо це розбиття через A_f.

Якщо задано три множини *M*, *R* і *S* і відображення $f : M \to R$ і $g : R \to S$, то суперпозиція даних відображень, тобто відображення $gf : M \to S$, де $(gf)(i) = g(f(i))$ задає розбиття A_{gf} яке складніше, ніж розбиття A_f.

Тепер візьмемо два розбиття множини $M : A = \{A_1, A_2, \ldots, A_k\}$ і $B = \{B_1, B_2, \ldots, B_l\}$. Розбиття $C = \{C_1, C_2, \ldots, C_m\}$ називається *добутком розбиттів* *A* і *B* і позначається $C = A \cdot B$, якщо воно являється подрібненням обох даних розбиттів, при чому найбільшим із всіх подрібнень.

На практиці часто доводиться розв'язувати задачі, які вимагають операції з натуральними числами і їх сумами. Зручним комбінаторним трактуванням для таких задач виявилося поняття розбиття числа.

Слід зазначити що існує добуток двох розбиттів.

Розглянемо зв'язок розбиттів і прямих добутків. Нехай задано дві множини: *A* з розбиттям A_i і *B* з розбиттям B_j. На їх добутку $C = A \times B$ легко визначається розбиття *C*, що складається із добутків $A_i \times B_j, A_i \in A, B_j \in B$, яке можна назвати добутком цих розбиттів. Дійсно, розбиття *A* і *B* легко перетво-

рюються в розбиття C_A і C_B множини C, тоді розбиття C являється добутком даних розбиттів: $C = C_A \cdot C_B$.

Визначення 1.4. Розбиттям натурального числа n є його представлення невпорядкованою сумою натуральних доданків: $n = n_1 + ... + n_r$, де доданки n_i називаються частинами, а їх число r – рангом розбиття.

Композиція – це представлення натурального числа n впорядкованою сумою натуральних доданків. Таким чином, композиції можна розглядати як «впорядковане розбиття».

В загальному вигляді екстремальні задачі на розбиттях можуть бути сформульовані у вигляді питання: наскільки багато існує розбиттів, що задаються в даній відповідності? Вибір конкретної відповідності між розбиттям визначається умовами практичної задачі, саме тієї, для якої розбиття з такою відповідністю служить комбінаторною схемою.

Далі будемо розглядати задачі, що моделюються екстремальними задачами на вище зазначених комбінаторних конфігураціях.

Висновки до розділу

При розв'язанні комбінаторних задач часто використовуються такі добре відомі конструкції з елементів скінченої множини, як сполучення, розміщення, перестановки й т. п. В розділі визначаються поняття цих конструкцій через формалізацію комбінаторних схем з використанням поняття відображення однієї скінченої множини X в іншу Y. На основі цього підходу побудовані комбінаторні конфігурації, які відповідають найпростішим комбінаторним об'єктам: розміщенням, перестановкам, комбінаціям, розбиттям та іншим об'єктам, а також розглянуті їх властивості, які в подальшому використовуються при розв'язуванні задач.

РОЗДІЛ 2. СТАН ПРОБЛЕМИ, НАПРЯМКИ ДОСЛІДЖЕНЬ ТА МЕТОДИ РОЗВ'ЯЗАННЯ ЕКСТРЕМАЛЬНИХ ЗАДАЧ ДИСКРЕТНОЇ ОПТИМІЗАЦІЇ

Екстремальні задачі дискретного характеру на комбінаторних конфігураціях постійно зустрічаються при розв'язуванні практичних задач. Проте довгий час вони не привертали до себе уваги, оскільки в більшості випадків для їх розв'язання знаходився деякий алгоритм розв'язування, зазвичай типу перебору. Широке застосування комп'ютерів істотно змінило положення, так як стало можливим розв'язувати екстремальні задачі великої розмірності. Виявилось, що для таких задач різні удосконалення природних алгоритмів можуть давати істотний виграш в часі роботи або в потребах необхідної пам'яті. Необхідність практичного розв'язання широкого кола екстремальних задач на комбінаторних конфігураціях привела до появи великої кількості методів, що ґрунтуються на природних алгоритмах, а у багатьох випадках і до побудови принципово нових методів, що враховують специфіку задачі. Це в свою чергу привело до необхідності порівняння якості теоретичних методів, а також дослідження загальних принципів побудови кращих комбінаторних алгоритмів.

2.1. Математична постановка задачі комбінаторної оптимізації та її властивості

У загальному випадку задачу оптимізації можна представити кортежем $< f, X, \Pi, D, ext >$, де $f : X \to R^1$ – задана цільова функція задачі, R^1 – числова пряма, X – простір розв'язків задачі, Π – предикат, який визначає підмножину $D \subseteq X$ допустимих варіантів розв'язку згідно наявних обмежуючих умов, $ext \in \{min, max\}$ – напрям оптимізації. У цих позначеннях задачу оптимізації можна переписати у наступному вигляді: необхідно знайти $x^* \in D \subseteq X$ таке, що

$$x^* = arg \min_{x \in D \subseteq X} f(x). \qquad (2.1)$$

В деяких роботах під задачею комбінаторної оптимізації розуміється проблема пошуку екстремумів заданої цільової

функції вигляду (2.1), коли X – комбінаторний простір. Під комбінаторним простором тут розуміється сукупність комбінаторних об'єктів певного типу, утворених із елементів заданої скінченної множини потужності n. Водночас, поняття «комбінаторний об'єкт» вживається як сполучення, перестановки, розміщення та ін. В зарубіжній літературі переважно вживається визначення задачі комбінаторної оптимізації, де простір варіантів X вважається скінченною множиною. Таке тлумачення не дозволяє строго формально окреслювати окремі класи задач комбінаторної оптимізації, такі, наприклад, як дискретне, цілочисельне чи булеве програмування. Більш того, воно часто призводить до фактичного ототожнення понять дискретної та комбінаторної оптимізації, яке і спостерігається в ряді робіт.

Найбільш вживане формулювання екстремальних комбінаторних задач можна визначити наступним чином: є n-множина елементів, на ній визначається скінченна множина комбінаторних конфігурацій $A=\{\pi_i\}=\{a_1, a_2, \ldots, a_q\}$, де $q > n$. Під комбінаторними конфігураціями $\pi_i = a_1, a_2, \ldots, a_q$ можна розуміти перестановки, розміщення, сполучення, розбиття, різні послідовності й т. п. На множині A задається функція $f(x)$. Потрібно відшукати екстремум $f(x)$ (максимум або мінімум) і елементи множини A, які цей екстремум доставляють.

Отже, задана деяка скінченна дискретна множина A. Комбінаторним простором $X = X_A$ назвемо сукупність усіх комбінаторних об'єктів визначеного вигляду з елементів множини A.

Тоді задачу (2.1) в загальному вигляді слід розуміти так: визначити точку x^* з простору X, що доставляє екстремум деякому функціоналу $f(x)$, тобто

$$f(x^*) = \underset{x \in X}{extr}\ f(x)$$

і задовольняє заданим умовам.

Саме формулювання екстремальних комбінаторних задач диктує широкий вибір операцій, які застосовуються для їх розв'язання. По-перше, треба вміти робити генерацію множини елементів A і мати у своєму розпорядженні відповідну множину

значень функції $f(x)$. По-друге, треба розробити методику порівняння цих значень і виділення з них максимального або мінімального. Операція переліку практично рідко виявляється здійсненною, тому що кількість можливих комбінацій може бути занадто великою. Труднощі, пов'язані з переліком варіантів і порівнянням значень, досить значні. Саме вони й були перешкодою для розвитку деяких розділів комбінаторного аналізу, незважаючи на їх очевидну актуальність. Лише застосування в математичній практиці обчислювальних машин створило можливості для розв'язання низки екстремальних задач.

Як відомо, більшість оптимізаційних задач комбінаторного типу можуть бути зведені до задач цілочисельного програмування. У роботі [203] встановлені деякі достатні умови зведення комбінаторних задач до задач цілочисельного лінійного програмування та показано, що не існує загального алгоритму, який знаходить таке зведення, навіть якщо воно існує. Але, зведення оптимізаційної комбінаторної задачі до вигляду оптимізаційної цілочисельної задачі не завжди доцільно, бо при цьому втрачаються комбінаторні властивості задачі та їх адекватний зміст [203, 206, 207].

Загальна схема зв'язку екстремальних комбінаторних задач з методами лінійного програмування виглядає приблизно так: елементи π_i інтерпретують, як точки евклідового простору, щоб «цільова» функція $f(x)$ стала лінійною формою. Розглядається задача знаходження екстремуму цієї функції на опуклій оболонці заданих точок (тобто на опуклому многограннику). Справді, екстремум лінійної форми на многограннику досягається в одній з вершин, які входять у множину розглянутих елементів. Задача ж знаходження екстремуму лінійної форми і є задачею лінійного програмування. Особливістю комбінаторних задач при такому зведенні залишиться те, що при знаходженні розв'язку варто обмежуватися лише точками із цілочисловими координатами.

Тоді розв'язок комбінаторних екстремальних задач являє собою набір $(a_1, a_2, \ldots, a_n)$ чисел $1, 2, \ldots, n$, множина яких представляє собою комбінаторну конфігурацію. Необхідно

знайти значення функції $f(x)=\sum_{i=1}^{n} c_i x_i$ на скінченній підмножині вибраних елементів конфігурацій.

Зазначимо, що коефіцієнти $c_1, c_2, \ldots, c_n$ цільової функції $f(x)$ як правило є невпорядкованими, але їх завжди можна впорядкувати за зростанням чи спаданням. Призначимо кожному коефіцієнту відповідний індекс з множини натуральних чисел $N_m = \{1, 2, \ldots, m\}$, тоді для деякої множини коефіцієнтів $C = (c_1, c_2, \ldots, c_n)$ розглянемо дворядковий запис відображення $\varphi : N \to C$:

$$\varphi = \begin{pmatrix} 1 & 2 & \ldots & m \\ \varphi(1) & \varphi(2) & \ldots & \varphi(m) \end{pmatrix}, \qquad (2.2)$$

де $\varphi(i) \in C, i \in N_m$. Упорядкований нижній рядок запису визначає коефіцієнт цільової функції і його місце в початковому векторі *C*. На відміну від елементів множини N_m, значення координат можуть співпадати і порядок розміщення різних значень координат суттєвий. Для подальшого викладу матеріалу введемо наступне

Визначення 2.2. Нормалізацією функції $f(x)=\sum_{i=1}^{n} c_i x_i$ назвемо операцію відображення перестановки $\varphi : N \to C$, що встановлює впорядкування коефіцієнтів $c_1, c_2, \ldots, c_n$ цільової функції за зростанням (спаданням).

Тоді функція є нормалізованою, якщо її коефіцієнти є впорядкованими. Зрозуміло, що для будь-якої функції $f(x)$ можна встановити порядок коефіцієнтів, тобто нормалізувати функцію. Щоб повернутися до попередньої початкової форми лінійної функції, необхідно зробити обернене відображення: $\varphi^{-1} : C \to N$.

Слід зазначити, що розв'язок, який одержується при зведенні і розв'язанні нової задачі методами цілочисельного лінійного програмування без врахування специфіки комбінаторної задачі є

не завжди адекватним до постановки вихідної задачі. Хоча існує ряд алгоритмів, що враховують для цілочисельної моделі задачі її комбінаторні властивості [52, 58–60, 98, 125, 216–221, 242, 243]. Таким чином, не існує чітко вираженої границі між оптимізаційними задачами, що представляються цілочисельними і комбінаторними моделями. Тому дуже важливо для розв'язання задачі підібрати відповідну модель задачі, що враховувала б найважливіші властивості й особливості задачі так, щоб врахування цих властивостей допомогло побудувати ефективні методи розв'язання задачі.

В іншому вигляді задачі визначені на просторі $X = X_A$ комбінаторних об'єктів, породжуваних деякою дискретною скінченною множиною *A*, можна інтерпретувати так: екстремум функціоналу $f(x)$, заданого на просторі *X*, визначається на всьому чи частині цього простору *R*, тобто $R \subseteq X$. Таким чином, оптимізаційну комбінаторну задачу в загальному вигляді можна записати таким способом: визначити $x^* \in X_A$ із співвідношення

$$f(x^*) = \underset{x \in R \subseteq X}{extr} f(x)$$

або у вигляді

$$x^* = arg\left(\underset{x \in R \subseteq X}{extr} f(x) \right).$$

Зрозуміло, що в силу скінченності множини *A*, оптимізаційна комбінаторна задача має розв'язок завжди, і цей розв'язок найчастіше неєдиний.

Таким чином, відпадає питання про дослідження задачі на можливість розв'язання (у випадку нескінченності множини *A* це питання потребує подальшого дослідження).

Однак проблема приймає форму практичної можливості розв'язання, тобто того, чи не перевищують запити даної оптимізаційної комбінаторної задачі наявні ресурси (швидкодія, пам'ять комп'ютера, потужність мікропроцесору та ін.). Отже, питання про практичну можливість розв'язання задачі залишається одним з основних питань обчислювальної математики.

Багато вчених в своїх працях [37, 58–60, 150–153] підкреслюють важливість комбінаторного представлення визначених класів дискретних оптимізаційних задач в різних інтерпретаціях, що дає можливість більш адекватно описати постановку прикладної моделі.

З погляду теорії графів, багато комбінаторних задач оптимізації мають наступне формулювання: відшукати серед деякої множини шляхів L мінімальний (або максимальний), тобто шлях, що володіє мінімальним (або максимальним) значенням λ. Як множина L може бути обрано, наприклад, множина всіх гамільтонових шляхів.

Слід зазначити, що за допомогою теорії графів добре описуються багато типів комбінаторних задач. При цьому графічні подання є не просто ілюстраціями, але й дозволяють одержувати нові підходи до їх розв'язання, а також результати.

Формальні постановки оптимізаційних комбінаторних задач можна розділити на два класи: безумовні та умовні. Ці поняття аналогічні відповідним поняттям з недискретної оптимізації. Умовність задачі найчастіше залежить від простору, у якому вона формулюється. Як правило, введення додаткових обмежень ускладнює задачу в алгоритмічному відношенні, однак зменшує область допустимих розв'язків, на якій розглядається задача.

Розглянемо надалі необхідні поняття комбінаторних конфігурацій, на яких розглядаються екстремальні задачі та їх властивості.

2.2. Моделі задач комбінаторної оптимізації

Новим теоретичним підходом для розв'язування задач проектування автоматизованих систем управління та інформаційних технологій є застосування методів комбінаторного аналізу, а саме, тематики екстремальних комбінаторних задач на різних комбінаторних конфігураціях. Високий ступінь абстракції постановок і розв'язання екстремальних комбінаторних задач дозволяє використовувати їх при проектуванні і технічних, і програмних засобів автоматизованих систем управління та при розв'язуванні різних прикладних задач економіки, фінансів, статистики та ін. Сукупність комбінаторних об'єктів утворює комбінаторні моделі, які на основі апріорної інформації про розв'язування прикладної задачі забезпечують повний опис всіх

їх показників та характеристик. Розглянемо огляд прикладних задач з [231, 232], що моделюються екстремальними задачами на комбінаторних конфігураціях. Зокрема, в даному пункті представлені як загально відомі дискретні оптимізаційні задачі, так і формалізовані порівняно недавно. Відомою класичною задачею на комбінаторній множині є задача про ранець, моделлю якої є цілочисельна лінійна модель. Також до загальновідомих задач комбінаторної оптимізації відноситься задача комівояжера. Як показує практика, найбільш вдалим для її розв'язання є алгоритм, що враховує її комбінаторні властивості.

2.2.1. Задача інтерполяції дискретної функції

Нехай на множині точок $A=\{a_1, a_2, \ldots, a_n\}$ експериментальним шляхом отримані значення деякої функції $f(a_1), f(a_2), \ldots, f(a_n)$. Треба визначити таку підмножину точок $x \subset A$, що містить не більше N елементів, $N < n$, щоб інтерполяція за цими точками функції $f(a)$ функцією $\psi(a)$, $a \in [a_1, a_n]$, визначеного вигляду давала мінімальне відхилення

$$\varphi(x) = \sum_{i=1}^{n} | f(a_i) - \psi(a_i) |,$$

тобто треба визначити $x^* \in M_A$, де M_A – множина, породжена елементами дискретної скінченної множини *A*, для якого

$$\varphi(x^*) = \min_{x \in R} \varphi(x),$$

де область *R* описана обмеженням $x^* \le N$.

2.2.2. Задача мінімізації часу розв'язування

Часто при розв'язанні задач в автоматизованих системах управління виявляється істотною черговість, у якій вони надходять для розв'язання в оперативну пам'ять обчислювальної системи; від цього може істотно залежати, наприклад, час розв'язання всіх задач множини *A* і низка інших показників ефективності організації обчислювальних процесів у системі. Останнє може виявитися важливим, наприклад, якщо задачі

взаємозалежні між собою, або коли розв'язання задач множини *A* здійснюється на великих схемах мультипрограмуванням. До такого роду задач комбінаторного типу зводяться також багато задач календарного планування (наприклад, коли розраховується черговість запуску у виробництво деталей, оброблюваних на декількох різнотипних верстатах відповідно до деякої технологічної програми). У таких випадках задачу доцільно формулювати в метричному просторі S_A розміщень.

Зокрема, нехай задача полягає в наступному. З даної множини *A* задач виділити підмножину потужності не меншу *D* й упорядкувати її так, щоб час розв'язання її на комп'ютері був мінімальним. Природно тут точку розв'язання задачі шукати серед елементів простору S_A.

2.2.3. Задача оптимізації ресурсів комп'ютера

Нехай задана множина $A=\{a_1, a_2, \ldots, a_n\}$ задач, що підлягають розв'язанню на комп'ютері. Функції $v=v(x)$ і $t=t(x)$ виражають відповідно пам'ять комп'ютера, необхідну для розв'язання підмножини задач $x \subset A$, і сумарний час їх розв'язання. Тоді вище зазначені функції виражаються залежностями

$$v(x)=\sum_{a_i \in x} v(a_i),\ t(x)=\sum_{a_i \in x} t(a_i),$$

які відповідно необхідно мінімізувати, тобто визначити $x^* \in M_A$, що задовольняє умовам $v(x) \le V$, $t(x) \le T$ і доставляє максимум функціоналу

$$f(x)=\sum_{a_i \in x} \varphi(a_i),$$

де $\varphi(a_i)=\alpha_1 v(a_i)+\alpha_2 t(a_i)$, *V*, *T*, α_1, α_2 – сталі.

2.2.4. Задача теорії розкладів

Одна із задач теорії розкладів для *m* машин полягає в наступному. Кожна з деталей a_i, що належать множині $A=\{a_i\}_{i=1}^{n}$, повинна послідовно пройти обробку на машинах $B_1, \ldots, B_m$.

Потрібно знайти x^*, що мінімізує деяку функцію $\Phi(x)$ за умови, що $T(x) \le T$, де $T = const$, $T(x)$ – час виконання плану x на машинах $B_1, B_2, \ldots, B_m$ відповідно до заданих програм обробки деталей.

Формальна постановка даної задачі може представлятися у двох просторах. Якщо не відомо, чи є величина T достатньою для того, щоб у шуканий план увійшли всі елементи множини A, то доцільно задачу розглядати в просторі розміщень S_A, тобто знаходити $x^* \in S_A$, мінімізуючи функцію $\Phi(x)$ за умови $T(x) \le T$.

Якщо ж відомо, що обмеження $T(x) \le T$ допускає розклад, що включає всі елементи множини A, то доцільно представляти задачу в просторі перестановок P_A. Таким чином, наявність додаткової інформації про потужності x^* дозволяє використати у формулюванні задачі простір меншої розмірності $P_A^* < S_A^*$, що позитивно позначається, загалом кажучи, на часі розв'язування задачі.

Зазначимо, що для практичного використання цієї постановки задачі необхідно ввести уточнення, що стосуються способу обчислення функцій $\Phi(x)$ і $T(x)$ відповідно до заданих програм обробки кожної з деталей $a_i \in A$, а також додатково ввести деякі інші обмеження типу: (на одній машині обробляється одна деталь та ін.).

2.2.5. Задача оптимізації оргструктури

Розглядається множина підприємств, зв'язки між якими фіксовані. Під зв'язками розуміються основні укрупнені показники, що характеризують матеріально-речовинні, фінансові й інформаційні точки. Ставиться задача оптимізації триступінчастої структури керування даною множиною підприємств.

Отже, задана сукупність $A = \{a_i\}_{i=1}^n$ підприємств і сукупність $B = \{a_i, a_j\}_{i,j=1}^n$ зв'язків між ними, $i \ne j$. Кожному підприємству a_i відповідає число $p(a_i)$, що вимірює обсяг виробництва.

Кожному зв'язку (a_i, a_j) відповідає число $f(a_i, a_j)$, що вимірює інтенсивність цього зв'язку. Необхідно знайти розбиття сукупності A на підмножини $x_1^*, x_2^*, \ldots, x_k^*$; кількість їх заздалегідь не фіксована й мінімізується сума взаємодій підмножин $x_1^*, x_2^*, \ldots, x_k^*$ при деяких додаткових умовах.

Цю задачу природно сформулювати в просторі розбиттів N_A. Нехай

$$X \in N_A \text{ і } X = \{x_1^*, x_2^*, \ldots, x_k^*\}.$$

Тоді сума взаємодій підмножин $x_1, x_2, \ldots, x_k$ визначиться таким способом:

$$F(X) = \sum_{r=1}^{k} \sum_{s=1}^{k} \sum_{a_i \in x_r} \sum_{a_j \in x_s} f(a_i, a_j).$$

Нехай, далі, c – обмеження на обсяг виробництва в одній підмножині; $\Delta = \left\| \delta(a_i, a_j) \right\|_{i,j=1}^{n}$ – матриця заборон,

$$\delta(a_i, a_j) = \begin{cases} 1, \textit{ якщо } a_i, a_j \textit{ допускається,} \\ \textit{розміщати в одне об'єднання,} \\ 0 - \textit{ в протилежному випадку.} \end{cases}$$

k – максимальне кількість підприємств, що дозволяється вміщати в одне об'єднання.

Таким чином, задача полягає в наступному.

Знайти елемент X^* простору розбиттів N_A, мінімізуючи функцію $F(X)$ на допустимій області $R \subset N_A$, тобто

$$F(X^*) = \min_{x \in R \subset N_A} F(X),$$

де елементи допустимої області R задовольняють перерахованим вище вимогам, а саме: якщо $X \in R \subset N_A$, $X = \{x_1, x_2, \ldots, x_k\}$, то для $s = 1, 2, \ldots, k$ маємо а) $\sum_{a_i \in x_s} p(a_i) \le c$; б) для всіх $a_i, a_j \in x_s$ $\delta(a_i, a_j) = 1$; в) $x_s \le k$.

2.2.6. Екстремальні задачі на розбиттях

А) Задача розбиття множини задач. Нехай, $\vartheta(x)$ позначає об'єм пам'яті комп'ютера, необхідний для розміщення програм задач із множини x.

Задача полягає в тому, щоб розбити задану множину задач на мінімальну кількість підмножин, що взаємно не

перетинаються, таким чином, щоб пам'ять комп'ютера, зайнята кожною із цих підмножин, не перевищувала заданого числа *V*. Аналогічні задачі розглядалися в роботі [155].

Комбінаторна модель цієї задачі представляється таким чином.

Знайти елемент X^* простору розбиттів N_A, що задовольняє умовам

$$\vartheta\left(x_i^*\right) \le V, \quad x_i^* \in X^*, \quad i = 1, 2, \ldots, X^*,$$

і такий, щоб $X^* = \min\limits_{X \in N_A} X$.

В) Задача дослідження процесу розподілу пам'яті комп'ютера.

Задачі на розбиттях чисел довгий час не виділялися як екстремальні, але в зв'язку з розвитком інформаційних технологій та необхідністю проектування автоматизованих систем управління для подолання проблем в цих галузях виникла потреба в розв'язуванні екстремальних комбінаторних задач на розбиттях.

Розглянемо одну з комбінаторних моделей, що дозволяють досліджувати процес розподілу оперативної пам'яті комп'ютера з сегментною організацією програм і даних [11].

Відомо, що у будь-який момент часу функціонування автоматизованих систем управління відбувається вплив зовнішньої фрагментації на процес розподілу пам'яті, і це достатньо повно характеризують наступні параметри:

- кількість вільних (зайнятих) ділянок пам'яті;
- розмір вільних (зайнятих) ділянок;
- сумарний розмір вільної (зайнятої) пам'яті.

Запити на виділення пам'яті в цих дослідженнях у будь-який момент часу функціонування автоматизованих систем управління достатньо повно характеризуються наступними параметрами:

– кількістю запитів в черзі на виділення пам'яті;

– необхідними розмірами неперервних ділянок адресного простору пам'яті або розмірами запитів;

– сумарним розміром пам'яті, потрібної для задоволення запитів в черзі.

Нехай Q – розмір оперативної пам'яті комп'ютера автоматизованої системи управління, а N – сумарний розмір вільної пам'яті, який в процесі функціонування системи приймає значення $N \in Z^+$, $N \le Q$, де Z^+ – множина цілих додатних чисел. Через вплив зовнішньої фрагментації пам'ять розміром N виявиться подрібненою на r вільних фрагментів, представлених ділянками неперервного адресного простору пам'яті. Такий стан вільної пам'яті можна інтерпретувати як вектор

$$z(N)(n_1,\ldots,n_r), N = \sum_{i=1}^{r} n_i,\ n_1 \ge n_2 \ge \ldots \ge n_r;$$

$N \in Z^+$ де n_i – розмір i-ї вільної ділянки пам'яті, а r – кількість таких ділянок.

При моделюванні запитів на виділення вільної пам'яті з $z(N) = (n_1, n_2, \ldots, n_r)$ припускаємо, що вони можуть надходити або одночасно, тобто групами $q(K) = (k_1, k_2, \ldots, k_t)$ або поодинці, де k_j – необхідний розмір вільної пам'яті для j-го запиту. Групу запитів також будемо тлумачити як вектор, тобто

$$q(K) = (k_1, k_2, \ldots, k_t), \sum_{j=1}^{t} k_j = K, k_1 \ge k_2 \ge \ldots \ge k_t,\ k_j \in Z^+.$$

Дві групи запитів вважатимемо різними, якщо вони різні як вектори.

Елементи n_i, d_l і k_j векторів $z(N)$, $g(D)$, $q(K)$ є натуральними числами. Отже, $z(N)$, $g(D)$, $q(K)$ можна тлумачити як розбиття чисел N, D, K відповідно, тобто $p(N) = (n_1, n_2, \ldots, n_r)$, $p(K) = (k_1, k_2, \ldots, k_t)$, $p(D) = (d_1, d_2, \ldots, d_l)$,

де частини розбиття n_i визначають розміри вільних ділянок адресного простору пам'яті, частини k_j – необхідні розміри неперервних ділянок адресного простору вільної пам'яті або розміри запитів на пам'ять, а частини d_m – розміри неперервних ділянок адресного простору зайнятої пам'яті. Ранги розбиття $p(N)$, $p(K)$, $p(D)$ визначають відповідно; r – число неперервних ділянок адресного простору вільної пам'яті, t – кількість запитів в черзі і l – число неперервних ділянок адресного простору зайнятої пам'яті.

Інтерпретація станів вільної і зайнятої пам'яті невпорядкованим розбиттям чисел дозволяє адекватно моделювати зовнішню фрагментацію пам'яті без врахування станів її адресного простору, що істотним чином спрощує проведення досліджень процесу розподілу пам'яті.

В задачі для подальших досліджень необхідно визначити: чи є в пам'яті вільні неперервні ділянки її адресного простору, необхідні для задоволення запитів, що надійшли в пам'ять? При такій постановці задачі не вимагається даних про стан адресного простору вільної пам'яті.

Представлення груп запитів на виділення пам'яті невпорядкованим розбиттям чисел також не протиставляється практичному сенсу досліджуваного процесу. Отже, розглянута модель є адекватним представленням як станів фрагментованої пам'яті, так і систем запитів на виділення пам'яті, які можуть утворюватися в процесі функціонування комп'ютера.

За допомогою множини розбиття чисел можна описати множину всіх можливих станів фрагментарної вільної пам'яті фіксованого розміру. Як вже наголошувалося, в процесі функціонування автоматизованої системи управління сумарний розмір вільної пам'яті комп'ютера змінюється в межах $0 \le N \le Q$, де Q − розмір пам'яті комп'ютера. Використовуючи наступну властивість множини розбиття чисел: $P(N_1) \bigcap P(N_2) \bigcap \ldots \bigcap P(N_r) = 0$, де $P(N_i)$ − множина розбиття числа N_i $\forall i \ne j, N_i \ne N_j$ можна показати, що множина станів фрагментарної вільної пам'яті комп'ютера розміром Q визначається множиною розбиття чисел

$$Z(Q) = \bigcup_{N=0}^{Q} Z(N) = \bigcup_{N=0}^{Q} \bigcup_{r=1}^{min(N,Q+1-N)} P_r(N).$$

Дані моделі досить наглядно представляють застосування екстремальних задач на розбиттях до розв'язування практичних проблем.

2.3. Сучасні методи розв'язання задач комбінаторної оптимізації

Методи розв'язування комбінаторних задач розвиваються досить інтенсивно. Очевидно, що швидко розповсюджуються методи, які краще і простіше враховують властивості і специфіку класів комбінаторних задач.

Загальні алгоритми для розв'язування екстремальних комбінаторних задач поділяються на дві категорії (в деякій мірі перетинні): алгоритми відтинаючої площини, що є розвитком симплекс-алгоритму, і перебірні алгоритми, засновані на розумному переборі всіх можливих розв'язків.

Основною ідеєю перебірних методів є перехід від повного перебору скінченної множини розв'язків до скороченого, направленого. Ці методи мають ряд позитивних якостей: гнучкість, універсальність, можливість застосування до різних задач комбінаторної оптимізації в будь-якій постановці. Найбільшого поширення набули методи комбінаторної оптимізації, що використовують методи гілок та меж [88], метод послідовного аналізу варіантів, методи побудови послідовності розв'язків, методи локальної оптимізації, метод динамічного програмування, апроксимаційно-комбінаторний метод [143] та ін. Евристичні алгоритми, що застосовуються при розв'язанні задач комбінаторної оптимізації, хоча досить швидко дають розв'язок, однак не гарантують його оптимальність.

2.3.1. Метод гілок та меж

Перші публікації про метод гілок і меж відносяться до початку 60-х років [230, 231]. У своїй роботі Ленд і Дойг метод гілок і меж застосований до оптимізаційних цілочисельних задач. Більш повно з врахуванням інших можливостей застосування і модифікацій описано метод гілок і меж в роботі Літтла, Мурті, Суїні, Керел [148, 149, 231] на прикладі задачі комівояжера. Тут же вперше вживається термін «метод гілок і

меж» і повідомляється про успішний чисельний експеримент для задачі комівояжера з 40 містами.

Як відомо, задача комівояжера формулюється в просторі перестановок як задача на безумовний мінімум. Маємо $(n-1)!$ (потужність простору перестановок з *n* елементів) циклів, один або декілька з яких дають мінімальну довжину. Метод повного перебору тут явно непридатний. Метод гілок і меж дав результати, кращі в порівнянні з усіма відомими методами, що і зумовило його популярність.

Алгоритм Літтла може бути застосований і для асиметричних задач, які з'являються в різних додатках. Автори [250–254] наводять приклад з області календарного планування.

Припустимо, що протягом деякого періоду часу складальна лінія повинна збирати вироби різних пакетів. Вартість переходу від виробу типу *i* до типу *j* дорівнює c_{ij}. Яка послідовність типів збираних виробів мінімізує сумарні витрати? Це задача комівояжера, в якій немає потреби вважати $c_{ij} = c_{ji}$.

Схема алгоритму така: множина всіх можливих циклів розбивається на все менші і менші підмножини, для кожної з яких обчислюють мінімальні можливості витрат (довжину) циклу або їх нижні межі. Відповідно до обчислених мінімальних значень меж циклу проводять послідовне подібне розбиття підмножин і врешті-решт визначають оптимальний цикл. Коли знайдена підмножина, що містить єдиний цикл, витрати у якого менше або рівні нижнім межам для всієї решти підмножин, то такий цикл - оптимальний.

Підмножини циклів зручно представляти як вершини дерева, а процес розбиття як галуження (поява нових гілок) дерева. Тому метод названий методом «гілок і меж».

Згідно з цим методом множину всіх допустимих комбінацій розбивають на підмножини $Q_1, Q_2, \ldots, Q_{s_1}$. Кожну з останніх потім розбивають на підмножини $Q_{i_1}, Q_{i_2}, \ldots, Q_{i_{s_1}}$, $(i = 1, 2, \ldots, s_1)$, і т. д., поки не дійдуть до окремих комбінацій. Цей процес галуження зручно зображати у вигляді дерева. На кожній підмножині $Q_{i_1, i_2, \ldots, i_s}$ визначають функцію переваги. Починаючи з вершини Q, пересуваються з етапу на етап, вибираючи кожного разу підмножину з мінімальним значенням цієї функції

(або максимальним, якщо цього вимагає умова задачі). Одержане дерево буде деревом рішень.

Задача полягає у відповідному виборі функції переваги, щоб одержати найкращий розв'язок. Іноді вибір підмножини відбувається випадковим чином: при цьому вірогідність вибору даної підмножини тим більше, чим менше значення його функції переваги.

Основна проблема в цьому методі - вибір способу визначення нижньої (або верхньої) межі. Знайти досить точне її значення не завжди легко, та зате число даних гілок на дереві рішень в загальному випадку скорочується. Це випливає з того, що якщо значення нижньої межі для якої-небудь підмножини більше або рівне (у разі мінімізації; менше або рівно – в протилежному випадку) значенню функції одного з вже одержаних рішень, що мінімізується, то відповідну гілку дерева рішень виключають з розгляду.

2.3.2. Послідовні алгоритми оптимізації

Метод послідовного виключення можливостей – один з найпоширеніших способів прийняття рішень. Зводиться він до наступного виконання кроків [231]:

а) множина альтернатив розбивається на підмножини шляхом введення додаткових висловлень, що характеризують кожну окрему підмножину;

б) частину цих підмножин прагнуть виключити, використовуючи умови несумісності (логічної суперечності) висловлень, що відносяться до елементів підмножини, і висловлень, що характеризують вимоги до розв'язку;

в) з множинами, що залишилися, роблять аналогічну процедуру розбиття і виключення і т. д.

На основі досвіду математичного використання принципу послідовного виключення можливостей при розв'язанні ряду оптимізаційних задач планування і проектування в Інституті кібернетики ім. В. М. Глушкова НАН України в 60-х роках минулого століття була розроблена формалізована схема прийняття рішень, що одержала назву методу послідовного аналізу варіантів.

У основі методу послідовного аналізу варіантів лежить ідея представлення процесу розв'язання у вигляді багатоступінчатої структури, що нагадує структуру складного досвіду. Кожен

ступінь пов'язаний з перевіркою наявності тих або інших властивостей у підмножини варіантів і веде або до безпосереднього скорочення початкової множини варіантів, або задає можливість такого скорочення в майбутньому. На основі теоретичного і практичного аналізу поставленої задачі спочатку потрібно чітко сформулювати, якими особливими властивостями повинен володіти шуканий варіант. Потім потрібно виявити по можливості більше ознак, які дозволяють встановити, що даний варіант не є шуканим. Серед цих ознак вибираються ті, що найлегше перевіряються і властиві одночасно, за можливістю, більшому числу варіантів. Після цього побудова числової схеми розв'язання полягає у виборі раціонального порядку перевірки ознак, що дозволяє в найкоротший час відсіяти непридатні варіанти і знайти оптимальний розв'язок.

Представлення процесу пошуку оптимального варіанту як послідовності складних дослідів нагадує формалізацію процесу ухвалення рішень на основі статистичних експериментів, розроблену А. Вальдом в теорії послідовних статистичних рішень.

У багатьох задачах для організації дослідів по звуженню множини можливих варіантів до шуканої множини вдається використовувати деякі загальні властивості оптимальних варіантів, що є узагальненнями принципу Р. Беллмана в динамічному програмуванні.

Загальна схема послідовного аналізу варіантів представляється таким чином.

Будується розв'язок багатоваріантної задачі як послідовний пошук достатньо звуженої підмножини варіантів на основі перевірки обмежень і обчислення критерію. Загальна схема такого пошуку може бути формалізована таким чином [231].

Нехай дано три множини: $W=\{\omega\}$ – множина варіантів, $\pi=\{\pi_\alpha\}$ – множина дослідів, $M=\{\alpha\}$ – множина індексів дослідів. У множині *M* виділена підмножина M^*, яку назвемо контрольною. Далі дана множина $I=\{\omega\}$, яку назвемо множиною результатів. Для кожного досліду π_α визначено в *I* підмножину $I_\alpha=\{\omega_\alpha^1,\omega_\alpha^2,\ldots\}$, кожен елемент якої назвемо результатом досліду π_α. У множині *I* виділена підмножина $\Omega=I$, на якій визначений оператор звуження $S(\omega)$, що ставить у

відповідність кожному $\omega \in \Omega$ деяку підмножину $W_\omega = S(\omega)W$ з W. Ця відповідність природним чином розповсюджується на підмножини U з множини W:

$$S(\omega)U = U \cap W_\omega = U \setminus V_\omega,$$

де $V_\omega = W \setminus W'_\omega$.

На множині дослідів π визначений оператор реалізації P, що ставить у відповідність кожному $\pi_\alpha \in \Pi$ деякий елемент з $I_\alpha : P_{\pi_\alpha} = \omega_\alpha^i$, який назвемо реалізацією досвіду π_α.

Задача полягає у визначенні такої максимальної підмножини W^* з W, яка є інваріантною щодо будь-якої з контрольної множини M^*:

$$S\left(P_{\pi_\alpha}\right)W^* = W^* \text{ для кожного } \alpha \in M^*.$$

Вважатимемо, що множина результатів I містить елемент e, який має особливе значення. Розв'язком початкової задачі буде множина W_ω, що є звуженням вихідної множини W.

2.3.3. Методи побудови послідовності розв'язків

Метод побудови послідовності розв'язків застосовний до задачі

$$f\left(x^*\right) = \min_{x \in R \subset X} f(x), \tag{2.4}$$

якщо [231]:

а) можна знайти скінченне розширення $R^0 \supseteq R$ множини допустимих розв'язків R і функцію-міноранту $g(x)$, визначену на R^0 таку, що $g(x) \le f(x)$ для всіх $x \subset R$;

б) можна побудувати алгоритм впорядкування φ, який на k-у кроці знаходить елемент $x_k \in R^0$, що має наступну властивість

$$g(x_k) = min\left\{g(x) \mid x \in R^0 \setminus \{x_1, \ldots, x_{k-1}\}\right\}. \tag{2.5}$$

Інакше кажучи, повинен існувати ефективний метод φ побудови послідовності $\{x_i\}$ розв'язків з R^0 у порядку неспадання міноранти $g(x)$.

Аналогічно для задачі максимізації $f(x)$ на множині R повинен існувати метод φ побудови послідовності $\{x_i\}$ розв'язків з R^0 у порядку незростання мажоранти $g(x)$, тобто такої функції, що $g(x) \geq f(x)$ для всіх $x \in R$.

Критерій оптимальності. Якщо існує таке натуральне число k, що $R_k = \{x_1, \ldots, x_k\} \cap R \neq \varnothing$, $f(x_k^*) = \min\limits_{x \in R_k} f(x) \leq g(x_k)$, то x_k^* – оптимальний розв'язок задачі (2.4).

Методи розв'язання конкретних задач дискретної оптимізації, побудовані по вище зазначеній схемі, полягають в знаходженні за допомогою алгоритму φ послідовності x_i розв'язків множини R у порядку неспадання міноранти для виконання критерію оптимальності (2.4). Ефективність таких методів залежить від вибору розширення R^0 і міноранти $g(x)$. Якщо не існує число k, що задовольняє критерію (2.5), то метод перетворюється на повний перебір.

Вкажемо деякі варіанти схеми [231].

1. Якщо $g(x) = f(x)$ для всіх $x \in R$, тобто немає необхідності апроксимувати цільову функцію, критерій (2.4) спрацьовує, як тільки знайдено допустимий розв'язок $x^k \in R$.

2. Якщо $R^0 = R$, тобто множина допустимих розв'язків не розширюється, то, будуючи лише допустимі розв'язки $\{x_i\}$, завжди можна оцінити відхилення кращого з одержаних допустимих розв'язків x_k^* від оптимального x^*. Дійсно, якщо на k-у кроці умова (2.4) не виконується, то числа $g(x_k)$ і $f(x_k^*)$ є межами мінімального значення цільової функції $f(x)$ на R. При цьому з кожним кроком алгоритму межі уточнюються. Як наближений розв'язок можна вибрати x_k^* – кращий допустимий розв'язок з одержаних. Оскільки $f(x_k^*) - f(x^*) \leq f(x_k^*) - g(x_k)$,

то максимальне відхилення наближеного розв'язку x_k^* від оптимального x^* відомо.

Модифікуючи умову б) загальної схеми методів побудови послідовності розв'язків, одержимо наступні схеми [231]:

3. Нехай має місце припущення:

б') існує алгоритм, що визначає множину розв'язків

$$X = \left\{ x \mid f^- \le g(x) \le f^+, x \in R^0 \right\},$$

не обов'язково впорядкованих за збільшенням міноранти. Тут f^-, f^+ – відповідно нижня і верхня межі для значень цільової функції на оптимальному розв'язку, наприклад $f^- = min\left\{ g(x) \mid x \in R^0 \right\}$. Оптимальний розв'язок визначається перебором розв'язків множини *X*. Ефективність даного варіанту залежить від числа елементів *X*, а точніше, від якості меж f^-, f^+.

4. Нехай має місце припущення:

б") існує алгоритм, який на *k*-у кроці виділяє множину розв'язків

$$H_k = \left\{ x \mid x \in R^0,\ f_k \le g(x) \le f_{k+1} \right\},$$

де f_k – зростаюча послідовність дійсних чисел.

Для послідовності $\{H_i\}$ критерій оптимальності приймає наступний вигляд: якщо існує таке натуральне число *k*, що

$$R \cap \bigcup_{i=1}^{k} H_i \neq \varnothing$$

$$f\left(x_k^*\right) = \min_{x_k} \left\{ f(x_k) \mid x \in R \cap \bigcup_{i=1}^{k} H_k \right\} \le f_{k+1},$$

то x_k^* – оптимальний розв'язок задачі (2.4).

Особливість цього варіанту схеми полягає у тому, що на *k*-у кроці дописується до послідовності не один розв'язок з R^0, а ціла група їх.

2.3.4. Методи відтинання для розв'язування комбінаторних задач

Серед точних методів розв'язування задач дискретної оптимізації широкого розвитку і розповсюдження одержав метод відтинання, ідея якого вперше була запропонована Данцигом, а потім була розвинута в багатьох роботах, зокрема в роботах Гоморі.

Ефективність методу відтинання знаходиться в прямій залежності від ефективності способу побудови відтинань. Реалізація різних підходів до побудови відтинань породила велику кількість методів цієї групи.

Для групи методів відтинаючих площин використовується ідея «регуляризації» задачі. Вона полягає в зануренні початкової дискретної області допустимих розв'язків у відповідну неперервну опуклу область, тобто в тимчасовому відкиданні умов дискретності. Далі до одержаної регулярної задачі застосовуються стандартні методи оптимізації. Слід зазначити, що ефективність методу відтинання знаходиться в прямій залежності від ефективності способу побудови відтинань, а це викликає певні складнощі. Різні підходи до побудови відтинань, і різні модифікації методу відтинань для задач комбінаторної оптимізації розглянуто в роботах [91, 94, 98].

Розглянемо загальний опис простого алгоритму відтинаючих площин, що відноситься до класу методів, корисних для задач середнього розміру і цікавих з теоретичної точки зору. Розглянемо задачу цілочисельного програмування в стандартній формі:

$$\min c' x,$$

$$Ax = b, \tag{2.6}$$

$x \geq 0$ цілочислові змінні, де A, b, c − цілочислові.

Запишемо відповідну задачу лінійного програмування без обмежень на цілочисельність

$$\min c' x,$$

$$Ax = b, \tag{2.7}$$

$x \geq 0$ послабивши умову цілочисельності задачі (2.6).

Припустимо, що отримано розв'язок послабленої задачі, наприклад, за допомогою симплекс-алгоритму і знайшли базисний допустимий розв'язок x^*. Звичайно, у загальному випадку x^* не цілочисловий. Проте можливо одержати розв'язок початкової задачі, округлюючи координати отриманого розв'язку x^* до найближчих цілих чисел. Рис. 2.1 показує, чому такий підхід не спрацьовує: насправді біля x^* може взагалі не бути допустимих точок.

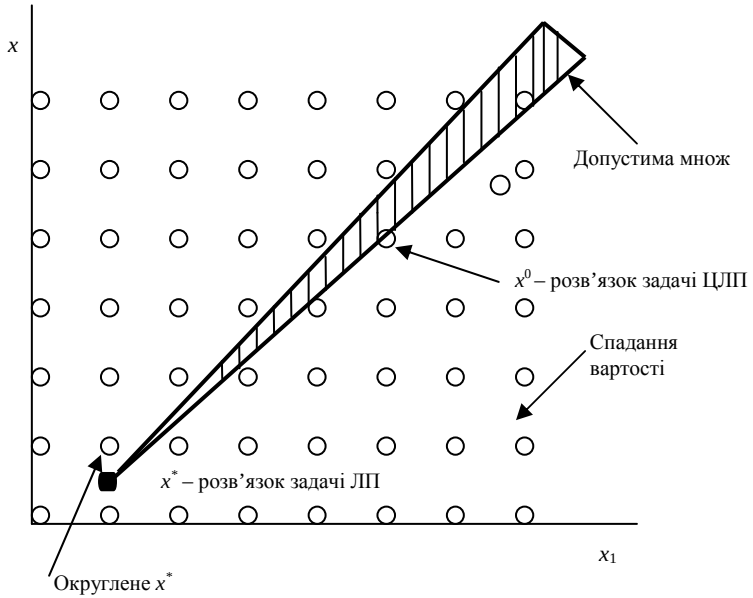


Рис. 2.1. Гіпотетична задача цілочислового лінійного програмування з оптимумом x^0 і її ослаблення з оптимумом x^*

Не важко бачити, що неперервний і дискретний оптимум розміщені досить далеко один від одного як за відстанню на площині, так і за оцінкою.

Слід зазначити два корисних факти. Перший: якщо неперервний оптимум x^* – розв'язок задачі лінійного програмування – виявляється цілим числом, то він розв'язує відповідну задачу цілочисельного лінійного програмування. Другий: оцінка $C\left(x^*\right)$

неперервного оптимуму є нижньою межею для $C\left(x^0\right)$ – оцінки дискретного оптимуму. Якщо до задачі цілочисельного лінійного програмування додати обмеження, що не виключає цілочислових допустимих точок, то розв'язок не змінюється. Отже, основна ідея алгоритму відтинаючих площин полягає в додаванні до задачі цілочисельного лінійного програмування таких лінійних обмежень по черзі, які б при розв'язанні відповідної задачі лінійного програмування, що є ослабленням початкової задачі, давали б цілочислові розв'язки.

Оскільки цілочислові допустимі точки не виключаються, то остаточний розв'язок ослабленої задачі цілочисельного лінійного програмування з доданими обмеженнями буде розв'язком початкової задачі. Описаний процес проілюстрований на рис. 2.2. На рис. 2.2(а) представлені початкова задача цілочислового лінійного програмування і неперервний оптимум x^*. На рис. 2.2(б) додане лінійне обмеження, що не виключає ніяких цілочислових допустимих точок, яке називається відтинаючою площиною (або просто відтинанням).

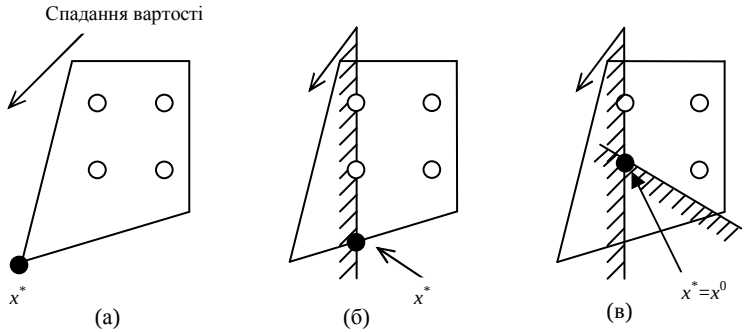


Рис. 2.2. Ілюстрація алгоритму відтинаючої площини:
а) неперервний оптимум x^*; б) новий розв'язок x^* після одного відтинання; в) розв'язок початкової задачі цілочисельного лінійного програмування після двох відтинань

При цьому відтинається частина допустимої множини, проте не втрачаються ніякі цілочислові точки. Новий неперервний оптимум зсувається, як показано на рисунку. На рис. 2.2(в) представлений результат додавання ще однієї відтинаючої

площини; в результаті неперервний оптимум виявляється цілочисловим, і ця точка є розв'язком початкової задачі цілочисельного лінійного програмування.

Метод відтинання для комбінаторних задач з лінійними функціями цілі на комбінаторних множинах розроблений в Полтаві під керівництвом доктора фізиком-математичних наук Ємця О. О. та апробований його учнями на різних комбінаторних множинах [89, 90]. Для комбінаторних задач суть даного методу полягає в наступному: 1) здійснюється релаксація задачі комбінаторної оптимізації: умова дискретності (належності розв'язку комбінаторній множині) замінюється умовою неперервності – належності опуклій оболонці комбінаторної множини, розглядається знову ж таки неперервна задача; 2) розв'язується симплекс-методом задача лінійного програмування на новій заданій області; 3) для отриманої точки перевіряється виконання всіх обмежень системи комбінаторного многогранника. Якщо точка задовольняє умову належності комбінаторній множині E, то задача розв'язана. Інакше здійснюється побудова відтинання, якому знайдена точка не задовольняє, у вигляді

$$\sum_{j=1}^{s} \alpha_{m+1,j} z_j \le b_{i+1},\ i \in J_p,$$

і знову розв'язується задача лінійного програмування, поки не буде знайдена точка, що є вершиною комбінаторного многогранника, а отже і елементом комбінаторної множини.

Треба зауважити, що область визначення комбінаторних задач – скінченна множина, що визначена обмеженнями комбінаторного многогранника, а тому замкнена, і тому через скінченну кількість кроків метод відтинання закінчує свою роботу, тобто знайдеться точка, яка є розв'язком вихідної задачі.

Слід зазначити, що ефективних і одночасно універсальних способів для розв'язання екстремальних оптимізаційних задач не існує. Але, навіть якщо б вони і існували, то не завжди витрати на пошук оптимального розв'язку виправдовують вигоди від його застосування, і часто в практичних ситуаціях віддають перевагу швидкому вибору розв'язку, що близький до оптимального. Тому разом з точними методами, що забезпечують справжній екстремум, розвиваються різноманітні

наближені методи, в яких пошук розв'язку тим або іншим способом обмежується.

2.3.5. Евристичні алгоритми

Особливий клас алгоритмів складають так звані евристичні алгоритми. Сюди входять наближені алгоритми, направлені на розв'язання однієї якої-небудь задачі.

Це пов'язано з тим, що точні методи оптимізації вимагають, як правило, дуже великого об'єму обчислень. Тому має велике практичне значення створення простих методів, що дають розв'язки, достатньо близькі до оптимальних. Методи, в яких не оцінюється близькість одержаних розв'язків до оптимальних, прийнято називати евристичними. Якщо ж можна оцінити відхилення розв'язків від оптимальних, то такий метод називають наближеним. Евристичний метод після його успішного теоретичного дослідження може перейти в розряд наближених.

Найчастіше в евристичних методах застосовують локальну оптимізацію.

Нехай маємо множину комбінацій $P = \{\pi_1, \pi_2, \ldots, \pi_N\}$; ставиться задача обчислення мінімуму функції, визначеної на цій множині, і відшукання комбінацій, на яких цей мінімум досягається. Метод локальної оптимізації полягає в наступному.

Для кожної комбінації $\pi_i \in P$ визначимо множину Q_i комбінацій, які називатимемо сусідніми з π_i. Сусідство розуміється в широкому значенні: так для множини перестановок сусідніми можуть вважатися дві з них, коди яких відрізняються однією транспозицією; для підмножини вершин цілочислової багатовимірної решітки сусідніми можуть вважатися дві вершини, що відрізняються на одиницю тільки в одній координаті і т. д. В термінах теорії графів це можна трактувати так: кожній комбінації $\pi_i \in P$ поставимо у відповідність деяку вершину *i* графа *G*. Сусідні вершини *i* та *j* графа *G* з'єднаємо ребром $(i, j)\left(\pi_j \in Q_i\right)$. Одержаний граф G_1 з множиною ребер *U* називатимемо графом сусідства комбінацій, або графом 1-сусідства, тобто сусідства на відстані одного ребра. У графі G_1 може існувати сукупність *Z* його вершин, в якій

$$F(\pi_i) = F(\pi_j), \text{ якщо } \pi_i, \pi_j \in Z$$

$$F(\pi_i) < F(\pi_j), \text{ якщо } \pi_i \in Z; \pi_j \notin Z; (i,j) \in U.$$

Таку множину Z назвемо ізольованою. Ця множина, зрозуміло, може складатися і з однієї вершини. Перший етап локального підходу, про який іде мова, полягає в тому, щоб, вибравши довільну комбінацію, визначити для неї граф сусідства, після чого на цьому графі для всіх $\pi_j \in Q_i$ визначити значення $F(\pi_j)$.

Другий етап полягає в операції, яку прийнято називати спуском. Знаходимо $\pi_{j_o} \in Q_i$ так, щоб

$$F(\pi_{j_0}) = \min_{\pi_j \in Q_i} F(\pi_j),$$

і якщо $F(\pi_{j_0}) < F(\pi_i)$, то переходимо до комбінації π_{j_o}.

Так за скінченну кількість кроків дійдемо ізольованої множини. Проте навіть попадання на ізольовану множину не дає упевненості, що досягнуто не локальний екстремум (в даному випадку, мінімум), а глобальний. Тому локальні «проби» доводиться продовжувати. Одним із способів продовження пошуків може бути вибір нової допустимої комбінації і повторення операції спуску в графі, який є об'єднанням нового графу сусідства і старого. Можливі і інші способи. Визначимо, наприклад, послідовність графів сусідства $G_1, G_2, \ldots, G_s$ (графи, у яких сусідство на відстані не більше 1, 2, …, s ребер). Після попадання в G_1 на ізольовану множину Z вершин переходимо до графа G_2 з довільної вершини $\pi_i \in Z$. Якщо Z і в G_2 залишається ізольованим, перейдемо до G_3 і т. д. Якщо ж з вершини π_i, можливий спуск в ізольовану множину Z_1, то, перейшовши в яку-небудь вершину $\pi_j \in Z_1$, повернемося до графа G_1 і повторимо процес вже з іншої вершини G_1.

У описаному підході є багато невизначеного. Особливо неясно, як будувати графи сусідства і чим ця побудова буде кращою за випадковий вибір комбінацій. Проте досвід підказує, що одержувати сусідні комбінації іноді легше, ніж будувати

випадкові послідовності. Аналіз конкретної задачі часто приводить до способу визначення сусідніх комбінацій, що забезпечує ефективний спуск до оптимального або достатньо кращого розв'язку.

Якщо допустима область R розв'язання задачі містить декілька точок, що реалізовують екстремум (мінімум або максимум) функції $f(x)$, то можна розглядати дві задачі оптимізації: визначення локального і глобального екстремумів. Перша з цих задач простіша, і часто її рішення практично прийнятне. Тому в даний час методи локальної оптимізації набули широкого поширення, особливо в застосуванні до складних оптимізаційних задач, тобто до задач великої розмірності, цільові функції і функції додаткових обмежень яких не мають властивостей, що дозволяють побудувати ефективні алгоритми визначення глобального екстремуму.

Локальний алгоритм може бути охарактеризований двома параметрами: величиною околу, що вивчається на кожному кроці, і числом ознак, що запам'ятовується про кожен елемент.

2.3.6. Метод вектора спаду

Метод вектора спаду є сімейством алгоритмів, об'єднаних загальною схемою. Ці алгоритми призначені для розв'язання дискретних оптимізаційних задач і дають локальний розв'язок [153, 155, 231].

Метод вектора спаду − ітераційний метод направленого перебору, що належить до типу градієнтних методів, розроблений і багатократно апробований в Інституті кібернетики ім. В. М. Глушкова НАН України під керівництвом академіка Сергієнка І. В. його учнями [226, 231].

Нехай X − деякий комбінаторний метричний простір з метрикою r. На підмножині $R \subseteq X$ визначена функція $f(x)$. Тоді загальна задача в таких позначеннях може бути сформульована таким чином. Визначити елемент $x \in R \subset X$, що доставляє екстремум функції.

У застосуванні до цієї задачі схема методу вектора спаду полягає в наступному.

1. Визначення початкового значення x_0; $x_i = x_0$.

2. Побудова множини $L_\rho(x_i) \cap R$, де $L_\rho(x_i)$ – окіл радіусу ρ з центром в точці x_i.

3. Розв'язок (точний або наближений) локальної задачі

$$f(x_{i+1}) = \underset{x \in L_\rho(x_i) \cap R}{ext} f(x).$$

4. Порівняння x_i і x_{i+1}. Якщо $x_i = x_{i+1}$, то обчислення припиняємо, приймаючи за локальний розв'язок задачі x_i; інакше переходимо на п. 2 алгоритму, замінюючи i на $i+1$ і т. д. до тих пір, поки на деякому кроці i^* не знайдеться $x_{i^*} = x_{i^*+1}$.

Ця схема є деякою сукупністю алгоритмів.

Дійсно: 1) $\rho = \rho(i)$; зокрема, можна прийняти $\rho = const$, допустима також багатозначність $\rho(i)$, що дозволяє уточнювати одержані розв'язки; 2) локальна задача може розв'язуватися різними методами.

Зазначимо, що ця схема в застосуванні до кожної конкретної задачі (класу задач) повинна враховувати її конкретні властивості (класу задач), тому метод в застосуванні до вужчого класу задач вимагає спеціальних додаткових досліджень, що враховують властивості простору R, його метрики і множини, а також властивості функції $f(x)$.

2.4. Екстремальні задачі оптимізації при умові багатокритеріальності та методи їх розв'язання

Результати дослідження практичних задач планування та керування показують, що в реальній постановці ці задачі є задачами з декількома цільовими функціями. Вираз «досягти максимального ефекту при найменших витратах» означає ухвалення рішення при двох критеріях. Оцінка діяльності підприємств і планування як системи прийняття рішень визначається на основі більше десятка критеріїв: виконання плану виробництва за об'ємом, за номенклатурою, плану реалізації, прибутку за показниками рентабельності, продуктивності праці і т. д. Отже, практично будь-яка прикладна задача є багатокритеріальною, яку звести до одного критерію, як правило, досить складно, оскільки цілей може бути значно більше. В цьому випадку

оптимізація проводиться за декількома частковими критеріями, і така задача в загальному випадку і називається задачею багатокритеріальної оптимізації.

Отже, багатокритеріальна задача це задача про оптимізацію двох і більше критеріїв, яка має реальний економічний зміст і відіграє важливу роль в економіці, плануванні виробництва та ін. Одним з важливих застосувань такої задачі може бути задача про досягнення мінімуму собівартості деякої продукції, визначення максимуму рентабельності виробництва, розрахунку прибутковості і взагалі про оптимізацію деяких відносних показників якості (трудомісткості, продуктивності і т. п.). Такі моделі відображають тенденцію постійного зниження рівня собівартості в розрахунку на одиницю продукції та підвищення якісних показників виробництва при збільшенні масштабів виробництва.

В зв'язку з цим особливе значення на даний час набуває теорія прийняття рішень при наявності багатьох критеріїв. Одним із основних понять теорії багатокритеріальної оптимізації є поняття оптимального за Парето, або ефективного розв'язку. Вперше проблема оптимізації векторного критерію була сформульована економістом Парето в 1896 р.

Проблема прийняття рішень в економіці, зокрема оптимального планування, виникає в силу двох принципових обставин: з одного боку, багатоваріантності планових рішень, з іншого − цілеспрямованості економічних систем. Множина альтернативних варіантів плану визначається наявними можливостями економічного розвитку; вибір із цієї множини визначається цілями планованої системи. Прийняте рішення є результатом спільного розгляду цілей і можливостей, їх узгодження одна з одною.

При використанні математичних методів в аналізі та прийнятті планових рішень обидві складові задачі вибору повинні знайти адекватне відображення в економіко-математичній моделі. Не зупиняючись тут на принципах опису множини допустимих варіантів плану, звернемося до методів математичного моделювання цілей економічного розвитку.

У найбільш простій і досить поширеній інтерпретації ціль розуміється як деякий заздалегідь визначений стан, досягнення якого задане в деякій розглянутій системі. Таке трактування, зрозуміло, досить обмежене, оскільки за його межами повністю

залишається задача вибору цілі, що складає зміст цільової стадії планування.

Отже, нехай елементи системи цілей представлені частковими критеріями $f_1,\ldots, f_l : X \to R;$ тут і далі X − множина допустимих планів, що ототожнюється зі своїм економіко-математичним описом і знаходиться у просторі R^n. Функції $f_1,\ldots, f_l$ формують векторний критерій оптимальності $F(x)=\left(f_1(x),\ldots, f_l(x)\right);$ вважається, що *j*-й цілі відповідає максимізація складової $f_j(x)$. Пара (F,X), утворена векторним критерієм $F : X \to R^l$ і множиною *X*, представляє собою модель, або задачу, багатокритеріальної (векторної) оптимізації.

Таким чином, векторний критерій оптимальності виступає як засіб для розв'язання існуючого протиріччя між складністю цілей економічних систем і обмеженими можливостями їх безпосереднього економіко-математичного моделювання. Разом з тим було б помилковим повністю пов'язувати виникнення багатокритеріальних задач зі специфікою математичного методу дослідження економічних проблем. Множинність цілей економічних систем має об'єктивний характер і знаходить своє модельне відображення у формі векторного критерію.

Часткові цілі, представлені компонентами векторного критерію, поєднані різного роду зв'язками. Зокрема, цілі пов'язані тим, що висувають вимоги до одних і тих же варіантів допустимих планів. Такі зв'язки опосередковані обмеженнями, що формують допустиму множину *X*, і умовно можуть бути названі внутрішніми.

У свою чергу, зовнішні зв'язки відображають порівняльну важливість цілей, їх нагальність, взаємозамінимість та ін. Ці зв'язки обумовлені структурою системи цілей, об'єктивно існуючими відношеннями між її елементами. Звичайно, зовнішні зв'язки опосередковані досить складними соціально-економічними взаємодіями й тому важко піддаються економіко-математичному опису. Незважаючи на те, що такі зв'язки є найважливішими реальностями задачі прийняття планових рішень, вони ніяк не відображені векторним критерієм і тому виходить, залишаються за межами багатокритеріальної моделі.

Тоді природно, що така модель, яка розглядається сама по собі, виявляється недостатньою для обґрунтованого вибору

оптимального варіанта плану. Часткові критерії $f_1,\ldots, f_l$, будучи в загальному випадку різнонаправленими, оптимізуються на множині X різними розв'язками рівнянь типу

$$x^j \in arg\ max\{ f_j(x) \mid x \in X \},\ j \in N_l.$$

Кожний із цих розв'язків може виявитися неоптимальним, а найчастіше просто незадовільним з погляду інших компонентів векторного критерію F. Отже, виникає характерна для багатокритеріальних задач проблема досягнення компромісу між частковими цілями або, говорячи інакше, узгодження цих цілей. Таке узгодження вимагає порівняння різних елементів системи цілей, їх зіставлення однієї з іншою, що неможливо без належного урахування всієї сукупності як внутрішніх, так і зовнішніх зв'язків між ними.

Отже, можна констатувати, що векторний критерій $F = (f_1,\ldots, f_l)$, навіть якщо його компоненти поставлені у відповідність всім складовим системи цілей, не є її еквівалентом і повинен бути посилений додатковою інформацією. До надходження такої інформації багатокритеріальна модель $Z(F,X)$ дозволяє робити висновок про порівняльну перевагу різних варіантів плану та про можливості вибору тих або інших альтернатив у тій мірі, у якій часткові критерії не суперечать один одному.

В багатокритеріальній задачі оптимізації порівняння розв'язків здійснюється за допомогою заданих на множині допустимих альтернатив числових функції $f_1,\ldots, f_s, f_{s+1},\ldots, f_m$, що називаються критеріями. Передбачається, що $m \geq 2$, бо при $m = 1$ задача оптимізації являється однокритеріальною.

В більшості випадках при дослідженні проблеми багатокритеріальності всі критерії, крім одного, обраного домінуючого, приймалися як обмеження, оптимізація проводилася за домінуючим критерієм. Такий підхід до розв'язання практичних задач значно знижує ефективність прийнятих рішень. У зв'язку із цим у даній роботі розглядається загальна постановка задачі багатокритеріальної (векторної) оптимізації і деякі основні підходи до її розв'язання при умові комбінаторних властивостей розв'язків.

В останні роки методам розв'язання багатокритеріальних задач присвячена велика кількість літератури та інтерес до цієї області продовжує зростати. Важливою особливістю сучасного етапу розвитку даного наукового напрямку є більш широке використання методів багатокритеріальної оптимізації для розв'язання практичних задач.

Методи багатокритеріальної оптимізації являють собою чисельну реалізацію певного правила вибору ефективної (Парето-оптимальної), слабо-ефективної (оптимальної за Слейтером), строго-ефективної (оптимальної за Смейлом) альтернатив. Вони класифікуються за типами інформації наступним чином [24]: методи багатокритеріальної оптимізації; методи, які не використовують інформацію про перевагу на множині критеріїв; методи, які використовують один тип інформації про перевагу на множині критеріїв; методи, які використовують різні типи інформації про перевагу на множині критеріїв; спеціальні методи.

Одним з найбільш розповсюджених методів розв'язання багатокритеріальних задач є метод зведення багатокритеріальної задачі до однокритеріальної шляхом згортання векторного критерію в суперкритерій. При цьому кожний критерій множиться на відповідний йому ваговий коефіцієнт (коефіцієнт важливості). Також серед відомих методів слід назвати такі як принцип справедливого компромісу, принцип слабкої оптимальності за Слейтером, принцип наближення за всіма локальними критеріями до ідеального розв'язку, метод квазіоптимізації локальних критеріїв (метод послідовних поступок).

Отже, в даному розділі розглянуто методи розв'язування екстремальних комбінаторних однокритеріальних та багатокритеріальних задач. Слід зазначити, що задачі багатокритеріальної оптимізації були розглянуті в багатьох роботах, але без врахування комбінаторних властивостей області допустимих значень, що не завжди адекватно описують моделі прикладних задач. Хоча, можна навести ряд прикладів, де множини – області допустимих розв'язків таких задач – мають переставні, сполучні та ін. властивості області допустимих значень, тобто ці задачі потрібно розглядати як екстремальні задачі на комбінаторних конфігураціях з багатьма критеріями. В наступних розділах розглядаються підходи до розв'язування задач комбінаторної оптимізації з багатьма критеріями на основі існуючих методів, а

також пропонуються нові підходи до розв'язання екстремальних комбінаторних задач.

Висновки до розділу

У даному розділі розглядаються екстремальні задачі дискретної оптимізації та методи їх розв'язання: робиться загальний огляд стану проблеми та напрямів досліджень. Проведено аналіз актуальності теми та важливість обраного напряму досліджень. Формулюється загальна постановка задачі та здійснюється огляд існуючих методів її розв'язання. Саме формулювання екстремальних комбінаторних задач диктує вибір операцій, залучених для їх розв'язання. Тому необхідно вміти робити генерування множини елементів комбінаторної конфігурації і мати у своєму розпорядженні відповідну множину значень функції $f(x)$. Важливим є методика порівняння значень функції і виділення з них максимального або мінімального. Операція переліку практично рідко виявляється здійсненною, тому що число можливих комбінацій може бути занадто великим. При розв'язанні комбінаторних задач часто використовуються такі добре відомі конструкції з елементів скінченної множини, як сполучення, розміщення, перестановки й т. п. В розділі визначаються поняття цих конструкцій через формалізацію комбінаторних схем з використанням поняття відображення однієї скінченної множини *X* в іншу *Y*. На основі цього підходу побудовані комбінаторні конфігурації, які відповідають найпростішим комбінаторним об'єктам: розміщенням, перестановкам, комбінаціям, розбиттям та іншим об'єктам, а також визначені їх властивості для подальшого використання при розв'язуванні задач.

РОЗДІЛ 3. МЕТОДИ ГЕНЕРУВАННЯ КОМБІНАТОРНИХ КОНФІГУРАЦІЙ ТА ПОБУДОВА ЇХ ГРАФІВ

В даному розділі пропонується новий метод розв'язування екстремальних задач на комбінаторних конфігураціях. Оскільки задачі розглядаються на різних комбінаторних конфігураціях, які відповідають найпростішим об'єктам: розміщенням, перестановкам, розбиттям, сполученням, то при їх розв'язуванні досить важливою є методика перебору елементів конфігурацій, тобто їх генерування та порівняння, а також дослідження всіх елементів деякого класу комбінаторних об'єктів. Комбінаторні алгоритми призначаються для виконання обчислень на дискретних скінченних математичних структурах, що представляють собою комбінаторні конфігурації. Слід зазначити, що існує велика кількість методів генерування комбінаторних об'єктів, в [148] розглядається три методи генерування таких комбінаторних об'єктів, як перестановки, розміщення, сполучення – лексикографічний, антилексикографічний, метод генерування за мінімальне число транспозицій. В усіх цих методах елементарною операцією, яка застосовується до елементів комбінаторної конфігурації є поелементна транспозиція, тобто обмін значеннями змінних. В [11] крім вище зазначених методів генерування розглядаються метод транспозиції суміжних елементів та метод заміни одного елемента іншим.

В розділі представлені нові методи генерування, що будують послідовності елементів конфігурацій з допомогою однієї транспозиції двох елементів (сусідніх чи не сусідніх), а також підхід, що грунтується на переміщенні максимального елемента множини, з якої утворюється конфігурація. В залежності від тих чи інших методів генерування елементів комбінаторних конфігурацій будуються підходи до розв'язування екстремальних задач оптимізації. Зокрема, представлено новий метод направленого структурування, що поєднує засоби комбінаторного аналізу і теорії графів. Основна ідея цього методу перекликається з відомим методом послідовного аналізу варіантів, що описаний в першому розділі, розглядаються загальні підходи побудови алгоритмів та ефективні методи розв'язування екстремальних задач на різних конфігураціях, серед яких найбільша увага приділяється методам роботи з великими масивами даних, алгоритмам генерування і сортування комбінаторних об'єктів, а також алгоритмам на графах, що отримали широкий розвиток та застосування.

Графи досить тісно пов'язані з поняттям комбінаторних конфігурацій: перестановок, розміщень, сполучень, розбиттів та ін. Випуклими оболонками таких комбінаторних конфігурацій є комбінаторні многогранники, властивості яких відіграють важливу роль для розв'язування задач оптимізації, тому розглянемо деякі поняття з теорії графів.

3.1. Використання теорії графів для розв'язування екстремальних комбінаторних задач

У термінах теорії графів формулюється велика кількість задач, пов'язаних з дискретними об'єктами [11, 62, 86, 87]. Такі задачі виникають під час проектування інтегральних схем, схем керування, у дослідженні автоматів, в економіці й статистиці, теорії розкладів і дискретній оптимізації. Очевидно, із всіх математичних об'єктів графи займають одне з перших місць як формальні моделі реальних систем. Графи знайшли застосування практично у всіх галузях наукових знань: фізиці, біології, хімії, математиці, історії, лінгвістиці, соціальних науках, техніці й т. п. Найбільшою популярністю теоретико-графові моделі користуються при дослідженні комунікаційних мереж, систем інформатики, хімічних і генетичних структур, електричних ланцюгів й інших систем мережної структури.

До типових задач теорії графів й їх додатків можна віднести наступні: задача про найкоротший ланцюг, заміна обладнання, складання розкладу руху транспортних засобів, розміщення пунктів швидкої допомоги, розміщення телефонних станцій, задача про максимальний потік, задача про розміщення та покриття, розфарбування в графах, зв'язність графів і мереж (проектування найкоротшої комунікаційної мережі, синтез структурно-надійної мережі циркуляційного зв'язку, аналіз надійності стохастичних мереж зв'язку), структурний синтез лінійних виробничих ланцюгів, покриття схеми заданим набором типових підсхем та ін.

Для екстремальних комбінаторних задач важливість застосування теорії графів пояснюється тим, що комбінаторні конфігурації, на яких представляються задачі, можна представити у вигляді графів.

Графи комбінаторних конфігурацій мають багато цікавих властивостей; при їх вивченні виникають задачі, що представ-

ляють інтерес не тільки для теорії графів, комбінаторики, топології і геометрії, але і для теорії лінійного програмування.

Використання властивостей графів комбінаторних конфігурацій можуть послужити підвищенню ефективності «традиційних» і розробці нових методів розв'язування екстремальних задач комбінаторної оптимізації. Комбінаторні моделі можуть бути застосовані для представлення оптимізаційних задач, що виникають при оптимальному розміщенні на графах. Теорія графів вивчає екстремальні властивості графів, розглядаючи множину його граней всіх розмірностей як деякий комплекс. Але при розв'язанні таких задач виникають проблеми, пов'язані з складністю математичних моделей, великим об'ємом інформації і т. д., оскільки більшість задач на комбінаторних конфігураціях є *NP*-повними. Більшість задач на графах стосується визначення компонент зв'язності, пошуку маршрутів, відстаней і т. п. Проте при розв'язанні прикладних задач відповідні їм графи, що залучаються, мають достатньо великий об'єм, а аналіз можливий лише із застосуванням сучасної обчислювальної техніки. В даному розділі досліджується екстремальна задача на комбінаторних конфігураціях. На підставі встановленого взаємозв'язку між задачами на комбінаторних множинах і графами многогранників комбінаторних множин вивчаються деякі структурні властивості допустимої області, а також сформульовано ряд тверджень, які дають можливість побудувати методи розв'язання комбінаторної задачі з використанням графів. Зокрема, в роботі [60] описаний спосіб побудови гамільтонового шляху усередині гіперграней. У статті [61] ставиться задача побудови гамільтонового шляху між гіперграними, тобто, як побудувати гамільтонів шлях, додавши до двох графів без спільних вершин третій граф, що не має спільних вершин з першими двома, якщо гамільтонів шлях побудовано для перших двох графів. Для подальшого викладу матеріалу сформулюємо деякі необхідні визначення і властивості.

Як було зазначено в попередньому розділі, досить важливим є комбінаторне представлення визначених класів дискретних оптимізаційних задач в різних інтерпретаціях, що дає можливість більш адекватно описати постановку прикладної моделі.

Отже, розглядаємо граф – як фігуру, яка складається з непорожньої скінченної множини V точок (вершин) і скінченної множини E орієнтованих чи неорієнтованих ліній (ребер), що з'єднують деякі пари вершин.

Нехай v_1, v_2 вершини графа, $e = (v_1, v_2)$ – ребро, що їх сполучає. Тоді вершина v_1 і ребро *e* інцидентні, ребро *e* і вершини v_1, v_2 також інцидентні. Два ребра, інцидентні одній вершині, є суміжними; дві вершини, інцидентні одному ребру, називаються також суміжними.

Надалі, якщо не вказано інше, вершини графа *G* позначатимемо буквою v з індексами чи без: v, v_2, v_{34}, множину вершин графа – *V;* ребра – буквою *e* з індексами чи без: e, e_6, e_{97}, відповідно множину ребер графа $G - E$. Ребро, що з'єднує деяку вершину саму з собою, називають петлею. Ребра, що з'єднують одну й ту саму пару вершин, називають мультиребрами. Граф, що не містить мультиребер та петель, називають звичайним графом. Граф, в якому допускаються мультиребра чи петлі, називають мультиграфом. Орієнтоване ребро $e = (v_1, v_2)$ називається дугою, яка виходить з вершини v_1 і заходить в вершину v_2.

Зазначимо, що в подальшому розглядаються графи, які не є мультиграфами, тобто такі графи що не містять петель.

Визначення 3.1. Граф, усі ребра якого неорієнтовані, називають неорієнтованим графом, а граф, усі ребра якого орієнтовані – орієнтованим графом, або орграфом.

Підграфом графа $G = \langle V, E \rangle$ називається граф G' для якого виконуються наступні умови $V' \subseteq V$ і $E' \subseteq E$, де V', V – множина вершин E', E – множина ребер.

Маршрутом у довільному графі, що починається у вершині v_1 і закінчується у вершині v_2, називають послідовність вершин та ребер вигляду:

$$v_1 e_{i_1} v_{i_1} e_{i_2} v_{i_2} e_{i_3} \dots v_{i_{n-1}} e_{i_n} v_2,$$

де всі сусідні елементи інцидентні.

Маршрут в неорієнтованому графі називається ланцюгом.

Шляхом в орієнтованому графі називається односторонній орієнтований маршрут.

Очевидно, що маршрут і для звичайних графів однозначно визначається послідовністю вершин: $v_1 v_{i_1} v_{i_2} \dots v_{i_{n-1}} v_2$.

Маршрут, який не містить повторень вершин і ребер, крім, можливо, двох крайніх вершин v_1 та v_2, називають простим.

Замкнений шлях (ланцюг) $(v_1 = v_2)$ називають контуром (циклом).

Для подальшого викладу підходу до розв'язання задачі розглянемо поняття про гамільтонові та напівгамільтонові шляхи.

Визначення 3.2. Гамільтоновим шляхом у графі називають простий шлях, який проходить всі вершини графа.

Розглянемо деякі прикладні задачі, що формулюються в термінах теорії графів.

Знаменита задача про комівояжера є комбінаторною задачею на множині перестановок і пов'язана із знаходженням гамільтонового циклу найкоротшої сумарної довжини. Як відомо, в цій задачі вважається, що є *n* міст, які неодмінно повинен відвідати комівояжер, – всі і рівно по одному разу, пройшовши загальний шлях найменшої сумарної протяжності. Якщо між містами є дороги, то вони інтерпретуються як ребра графа порядку *n* з вказаною довжиною. Вершини такого графа – вершини переставного многогранника, і є містами.

Задача комівояжера є приклад комбінаторної задачі на графах і має велике практичне і теоретичне значення. Через свою обчислювальну складність вона рівносильна цілому класу перебірних задач і часто використовується математиками для порівняльного аналізу і вивчення складності алгоритмів оптимізації в дискретній математиці.

Деякі прикладні задачі, що формулюються в термінах теорії графів представлено в [140, 177].

Задача 3.1. Нехай задано *N* міст, між кожною парою міст необхідно забезпечити автомобільне сполучення. Ця мета досягається завдяки побудові мережі доріг між деякими безпосередньо близькими містами. Вартість будівництва доріг відома, необхідно для досягнення мети витратити мінімум ресурсів. Задача зводиться до пошуку мінімального остовного дерева, що визначається вартістю будівництва. Та ж задача може бути інтерпретована, як прокладання комп'ютерної мережі, при мінімальній витраті кабелів.

Задача 3.2. Припустимо, необхідно найняти людей, які виконують певну роботу. Кожна людина може працювати в різний час, і брати за цю роботу різну кількість грошей. Необхідно найняти працівників так, щоб від моменту А до моменту Б хоча

б один з працівників виконував цю роботу, і витратити на це мінімум грошей. Це задача на визначення найкоротшого шляху. Для її розв'язання необхідно побудувати мультиграф. Вершини – це моменти від А до Б тобто час, в який працівник може заступити на роботу і піти з неї, а ребра – це різні працівники Необхідно відшукати мінімальний шлях від А до Б. Причому, граф орієнтований.

Задача 3.3. (про зайнятість працівників) Нехай є n працівників і m видів робіт. Кожен працівник може виконувати якусь роботу, і кожну роботу можуть виконувати різні люди. Необхідно розподілити робітників так, щоб максимальна кількість була зайнята, причому одну роботу може виконувати один робітник. Будуємо граф, де певного робітника сполучаємо з деяким видом робіт, які він може виконувати, також створюємо псевдоджерело, і сполучаємо його із усіма працівниками. Всі роботи сполучаємо з псевдостоком. Вага дуг, що сполучають працівника і роботу рівна 1. Тоді максимальна кількість зайнятих людей рівна пропускній спроможності графа, який носить назву мережі.

Також можуть бути сформульовані задачі складання розкладу, аналізу мереж в електротехніці, аналізу ланцюгів Маркова в теорії ймовірностей, в програмуванні, в проектуванні електронних схем, в економіці, в соціології і т. ін. Досить важливим при розв'язуванні вище сформульованих задач є властивості множини вершин графа, а також співвідношення між графами та підграфами.

Надалі поняття теорії графів використаємо для представлення графів комбінаторних конфігурацій, граф комбінаторної конфігурації – це головний об'єкт, який надалі буде використовуватися для побудови нового методу направленого структурування.

3.2. Методи генерування комбінаторних конфігурацій

Класичною задачею комбінаторики є задача визначення числа способів розміщення деяких об'єктів в якійсь кількості «ящиків» так, щоб були виконані задані обмеження. Цю задачу можна сформулювати більш формально таким чином [148]: дані множини X, Y, причому $|X| = n$, $|Y| = m$, визначити скільки існує функцій $f : X \to Y$, що задовольняють заданим обмеженням?

Елементи множини X відповідають об'єктам, елементи множини Y – ящикам, а кожна функція $f: X \to Y$ визначає деяке розміщення, указуючи для кожного об'єкту $x \in X$ ящик $f(x) \in Y$, в якому даний об'єкт знаходиться.

Без втрати загальності можемо завжди вважати, що $X = 1,\ldots,n$ і $Y = 1,\ldots,m$. Кожну функцію f можна тоді ототожнити з послідовністю $\langle f(1),\ldots, f(n)\rangle$.

Якщо $|X| = n$, $|Y| = m$, то число всіх функцій $f: X \to Y$ дорівнює m^n і таким чином визначається множина розміщень.

Легко знайти кількість розміщень, для яких кожен ящик вміщує не більше одного об'єкту – такі розміщення відповідають взаємно однозначним функціям. Позначимо $[n]_m$ число всіх взаємно однозначних функцій з n-елементної множини в m-елементну множину. Легко показати, що

$$[n]_m = n(n-1)\ldots(n-m+1). \qquad (3.1)$$

Якщо $m = n$, то кожна взаємнооднозначна функція $f: X \to Y$ є взаємно однозначним відображенням множини X на множину Y. У такому разі $[n]_n = n(n-1)\cdot\ldots\cdot 1$ позначаємо $n!$. Кожне взаємнооднозначне відображення $f: X \to X$ називається перестановкою множини X.

В більшості комбінаторних методів для їх подальшої реалізації і застосування до розв'язування задач є важливим встановлення початкових умов тобто, множини комбінаторних об'єктів, що задовольняють деяким властивостям. Для цього є потреба в генеруванні і створенні таких комбінаторних об'єктів.

В попередньому розділі було розглянуто властивості комбінаторних множин як об'єктів – підмножин заданої дискретної множини. На основі даних властивостей можна формувати і розглядати елементи комбінаторних множин як вершини графів та деяких многогранників. Далі розглянемо відомі методи генерування комбінаторних множин [11, 148] та опишемо нові, для подальшого їх використання в комбінаторних методах. Для початку, як приклад, розглянемо перестановки, що найчастіше використовуються при моделюванні прикладних задач.

3.2.1. Методи генерування перестановок

Розглянемо перестановку як впорядковану множину з n елементів, які різняться між собою порядком їх розміщення. Число перестановок множини $\{a_1, a_2, \ldots, a_n\}$ різних елементів дорівнює $n!$. Відомо, що такі перестановки ізоморфні перестановкам індексів цих елементів. Тому будемо вважати, що елементами множини перестановок є числа від 1 до n.

Таке спрощення буде можливо також провести і для інших комбінаторних об'єктів, про які буде далі йти мова.

Розглянемо для подальшого використання та розв'язування задач три відомих методи генерування послідовності всіх $n!$ перестановок n-елементної множини, що описані в [11, 148, 184].

Лексикографічний метод формування перестановок. Послідовність перестановок на множині $\{1, 2, \ldots, n\}$ представлена в лексикографічному порядку, якщо вона записана в порядку зростання чисел. В загальному випадку якщо $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ і $\tau = (\tau_1, \tau_2, \ldots, \tau_n)$ – перестановки, то вважаємо, що σ лексикографічно менше τ, тоді і тільки тоді, якщо для деякого $k > 1$ маємо $\sigma_j = \tau_j$ для всіх $j < k$ і $\sigma_k < \tau_k$. Легко точно визначити відповідність між цілими числами $0, 1, 2, \ldots, n!-1, n!$ та перестановками із множини $\{1, 2, \ldots, n\}$, записаними в лексикографічному порядку. Позначимо перестановку $p = (\pi_1, \pi_2, \ldots, \pi_n)$.

Перестановки можна породжувати одну за іншою таким чином: починаючи з початкової перестановки $p = (\pi_1, \pi_2, \ldots, \pi_n)$ від неї переходимо до іншої шляхом перегляду p справа наліво у пошуку найправішої позиції, в якій $\pi_i < \pi_{i+1}$. Знайшовши її, тепер шукаємо π_j, найменший елемент, розташований праворуч від π_i і більший за нього; потім здійснюється транспозиція елементів π_j і π_i і відрізок $\pi_{i+1}, \ldots, \pi_n$ (елементи якого розташовані в порядку спадання) перевертається. Таким чином одержуємо нову перестановку, з якою також виконуємо вище зазначені дії.

Загальну ефективність алгоритму за лексикографічним впорядкуванням, що означає запис слів в алфавіті, визначають числом порівнянь між елементами перестановки.

Зазначимо, що в лексикографічно впорядкованій послідовності перестановок існує повна підпослідовність $k!$ перестановок із k правих елементів, в якій решта елементів не переміщується. Це спостереження дозволяє отримати рекурентне співвідношення для числа транспозицій $\pi_r \leftrightarrow \pi_s$ або $\pi_i \leftrightarrow \pi_j$ і числа порівнянь $\pi_i > \pi_{i+1}$ або $\pi_1 > \pi_j$ відповідно використаних для породження перших $k!$ із $n!$ перестановок.

Зокрема, зазначимо, що для кожного із n можливих значень π_1 тобто для кожної підпослідовності перестановок, відповідних $n-1$ найправішим компонентам, використовується I_{n-1} транспозицій. Трансформація останньої перестановки однієї з цих підпослідовностей в першу перестановку наступної підпослідовності вимагає $\lceil (n+1)/2 \rceil$ перестановок, і всього таких перетворень буде $n-1$. Таким чином, за винятком перетворення, яке здійснюється при $i=0$, існує $nI_{n-1}+(n-1)\lceil (n+1)/2 \rceil$ транспозицій, а тому

$$I_n = nI_{n-1} + (n-1)\left\lceil \frac{n+1}{2} \right\rceil \text{ і } I_1 = 0 .$$

Слід зазначити, що лексикографічний порядок не може породжуватися так ефективно, як інші порядки, перетворення однієї перестановки в іншу в цьому випадку – це складний процес. Важливість лексикографічного порядку визначається його простотою і природністю.

Розглянемо ефективність цього лексикографічного породження перестановок у вигляді алгоритму, для цього представимо перестановки в вигляді масивів. Припустимо, що елементи множини перестановок запам'ятовуються у вигляді елементів масиву $P = \{P[1], \ldots, P[n]\}$. Елементарною операцією, яка застосовується до масиву P є поелементна транспозиція, тобто обмін значеннями змінних $P[i]$ і $P[j]$ де $1 \le i, j \le n$. Ця операція згідно методів математичного програмування позначається через $P[i] :=: P[j]$. Очевидно, що вона еквівалентна послідовності команд

$$d := P[i]; P[i] := P[j]; P[j] := d,$$

де d – деяка допоміжна змінна.

Тоді метод лексикографічного впорядкування легко зрозуміти, якщо елементами, які переставляються, взяти числа $1,2,\ldots,n$. На множині всіх послідовностей довжини n з елементами з множини $X = \{1,\ldots,n\}$ визначається лексикографічний порядок:

$$\langle x_1, x_2, ..., x_n \rangle < \langle y_1, y_2, ..., y_n \rangle \Leftrightarrow \exists k \geq 1 \left(x_k \leq y_k \wedge (x_i = y_i) \forall i < k \right).$$

Зазначимо, що якщо замість чисел $1,2,\ldots,n$ взяти букви $a,b,\ldots,z$, то лексикографічний порядок визначає стандартну послідовність, в якій слова довжини n з'являються в словнику.

Антилексикографічний порядок визначається подібним чином до лексикографічного порядку $<'$, з тією різницею, що як черговість позицій в послідовності, так і впорядкування елементів множини X зворотні по відношенню до початкових даних, що визначені згідно лексикографічного порядку:

$$\langle x_1, x_2, \ldots, x_n \rangle <' \langle y_1, y_2, \ldots, y_n \rangle \Leftrightarrow \exists k \geq 1 \left(x_k > y_k \wedge (x_i = y_i) \forall i < k \right).$$

Як приклад розглянемо перестановки множини $X = \{1,2,3\}$ в лексикографічному (а) і антилексикографічному (б) порядку:

(a)			(b)		
1	2	3	1	2	3
1	3	2	2	1	3
2	1	3	1	3	2
2	3	1	3	1	2
3	1	2	2	3	1
3	2	1	3	2	1

Рис. 3.1. Перестановка множини

Алгоритм генерування перестановок в антилексикографічному порядку сформулювати набагато зручніше: 1) спочатку

кожну перестановку лексикографічного порядку записати в зворотному вигляді; 2) всі перестановки записати в оберненому порядку від $n!$ до 1. Відмітимо, що послідовність перестановок множини $\{1,\ldots,n\}$ в цьому випадку має наступні властивості:

1) в першій перестановці елементи йдуть в тій послідовності, що зростає, а в останній – в спадаючій; іншими словами, остання перестановка є протилежна першій;

2) у останній позиції, визначають послідовність перестановок множини $\{1,\ldots,n\}\setminus\{p\}$ в антилексикографічному порядку.

Зазначимо, що число транспозицій при генеруванні кожної наступної перестановки є величина змінна: кожна друга перестановка вийшла за рахунок однієї транспозиції $P[i] =: P[j]$, але разом з ними є і такі, які вимагають $\lfloor (n-1)/2 \rfloor + 1 = \lfloor (n+1)/2 \rfloor$ транспозицій. Хоча середнє число транспозицій, що припадає на кожну перестановку, невелике, проте в деяких випадках кращим був би алгоритм, в якому кожна наступна перестановка утворюється з попередньої за допомогою виконання однієї транспозиції. Це досить суттєво, коли з кожною перестановкою пов'язані деякі обчислення і коли існує можливість використання часткових результатів, одержаних для попередньої перестановки, якщо послідовні перестановки мало відрізняються одна від одної.

Основну схему алгоритму можна описати за допомогою наступної процедури, що полягає в генеруванні всіх елементів перестановок $P[1],\ldots,P[n]$ через послідовну генерацію всіх елементів перестановок $P[1],\ldots,P[n-1]$ і заміну елементу $P[n]$ на один з елементів $P[1],\ldots,P[n-1]$.

Розглянемо метод генерування всіх перестанонок за мінімальне число транспозицій. Даний метод генерування перестановок будує послідовність, в якій різниця між двома послідовними перестановками ще менше ніж в двох попередніх: кожна наступна утворюється з попередньої за допомогою одноразової транспозиції сусідніх елементів [148].

$(а)$				$(б)$				$(в)$			
1	2	3	4	1	2	3	4	1	2	3	4
2	1	3	4	2	1	3	4	2	1	3	4
1	3	2	4	2	3	1	4	2	3	1	4
3	1	2	4	3	2	1	4	2	3	4	1
2	3	1	4	3	1	2	4	3	2	4	1
3	2	1	4	1	3	2	4	3	2	1	4
1	2	4	3	4	3	2	1	3	1	2	4
2	1	4	3	3	4	2	1	1	3	2	4
1	4	2	3	3	2	4	1	1	3	4	2
4	1	2	3	2	3	4	1	3	1	2	4
2	4	1	3	2	4	3	1	3	4	1	2
4	2	1	3	4	2	3	1	3	4	2	1
1	3	4	2	4	1	3	2	4	3	2	1
3	1	4	2	1	4	3	2	4	3	1	2
1	4	3	2	1	3	4	2	4	1	3	2
4	1	3	2	3	1	4	2	1	4	3	2
3	4	1	2	3	4	1	2	1	4	2	3
4	3	1	2	4	3	1	2	4	1	2	3
2	3	4	1	4	2	1	3	4	2	1	3
3	2	4	1	2	4	1	3	4	2	3	1
2	4	3	1	2	1	4	3	2	4	3	1
4	2	3	1	1	2	4	3	2	4	1	3
3	4	2	1	1	4	2	3	2	1	4	3
4	3	2	1	4	1	2	3	1	2	4	3

Рис. 3.2. Послідовності перестановок, що одержані різними способами

На рис. 3.2 послідовності перестановок, одержані за допомогою методів: а) лексикографічний; б) антилексикогра-фічний; в) метод генерування за мінімальне число транспозицій.

Даний метод приписується Джонсону і Троттеру [148]. Суть його розглянемо на прикладі. Припустимо, що вже побудована послідовність перестановок елементів $2,3,\ldots,n$, що має властивість згідно методу, наприклад: 23, 32 для $n=3$. Тоді необхідну послідовність перестановок елементів $1,2,\ldots,n$ одержимо, вставляючи елемент 1 всіма можливими способами в кожну перестановку елементів $2,3,\ldots,n$. Одержимо:

$$\{1\,2\,3\},\{2\,1\,3\},\{2\,3\,1\},\{3\,2\,1\},\{3\,1\,2\},\{1\,3\,2\}.$$

У загальному вигляді елемент 1 розташовується між першою і останньою позиціями по черзі вперед і назад $(n-1)!$ раз. На основі цієї конструкції можна легко одержати рекурсивний алгоритм, що генерує необхідну послідовність перестановок для довільного *n*. Але слід зазначити, що даний метод має недолік: послідовність перестановок будується разом вся і лише після закінчення побудови всієї послідовності елементів перестановки, її можна зчитувати, а тому розв'язування таким способом потребує великого об'єму пам'яті. Розглянемо нерекурсивний варіант цього алгоритму. У цьому варіанті для кожного $i, 1\le i\le n$, булева змінна $PR[i]$ містить інформацію про те, чи переноситься елемент i вперед ($PR[i]$ – істина) або ж назад ($PR[i]$ – хибна), змінна $C[i]$ показує, яку з можливих $n-i-1$ позицій елемент 1 займає щодо елементів $i+1$, …, n на своєму шляху вперед або назад. Позицію елементу 1 визначаємо на підставі його позиції в блоці, що містить $i,i+1,\ldots,n$, а також на підставі числа елементів з $1,2,\ldots,i-1$, які знаходяться зліва від цього блоку. Це число, буде значенням змінної *x*, обчислюється як число елементів $j<i$, які б рухаючись назад, досягли б крайнього лівого положення. Кожна нова перестановка утворюється транспозицією найменшого із елементів *j*, який не знаходиться в граничному положенні з його лівим або правим сусідом.

Інтерпретацію послідовностей, одержаних за допомогою вище описаних методів генерування перестановок розглянемо за допомогою графа G_n. Вершини графа G_n можна розглядати як

перестановки множини $\{1,\ldots,n\}$, в якому дві вершини, відповідних перестановок f і g, сполучені ребром, тоді і тільки тоді, коли g утворюється з f одноразовою транспозицією сусідніх елементів. Якщо графу G_n надати орієнтацію, визначивши орієнтовані ребра та з множини вершин – набір упорядкованих пар різних вершин, то в графі G_n можна визначити шлях – маршрут, в якого всі вершини різні.

Для перестановок можна побудувати замкнутий маршрут, який проходить через кожну перестановку один раз і визначає гамільтонів шлях.

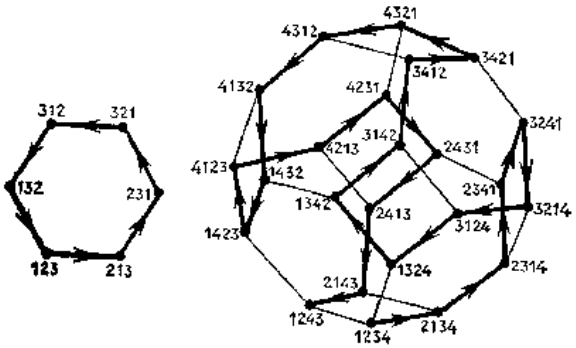


Рис. 3.3. Інтерпретація послідовності перестановок, одержаної за допомогою методу генерування переміщення за мінімальне число транспозицій

На рис. 3.3 стрілки вказують напрямок і послідовність обходу по вершинах графа – перестановках, що одержана за допомогою методу генерування всіх перестанонок за мінімальне число транспозицій. Дана послідовність відповідає гамільтоновому шляху в графі.

Вище описано огляд відомих методів генерування множини перестановок, кожний з яких має як свої переваги так і свої недоліки. Надалі визначимо нові методи генерування комбінаторних конфігурацій, які краще враховують специфіку розглядуваних задач і позбавлені тих недоліків, що мають відомі методи, а також дають можливість побудувати і представити граф комбінаторної конфігурації, який можна орієнтувати за значеннями цільвої функції.

Рекурсивний метод побудови перестановок – полягає в тому, що один елемент фіксується в кінці перестановки, а останні перебираються. Тобто, нехай маємо послідовність чисел $P=(p_1, p_2, \ldots, p_n)$, для якої виконується умова $p_1 \le p_2 \le \ldots \le p_n$. Тоді на першому етапі фіксується останній елемент – p_n, всі інші $(n-1)$-елементів $p_1, p_2, \ldots, p_{n-1}$ перебираються рефлексивно. Наступний крок полягає в тому, що елемент p_{n-1} поміщається на останню позицію і фіксується, а набір $(n-1)$ елементів послідовності *P*, що складається з $p_1, p_2, \ldots, p_{n-2}, p_n$ переставляється. Для подальшого розуміння знову замість змінних елементів $p_1, p_2, \ldots, p_n$ візьмемо числа $1, 2, \ldots, n$, множину яких позначимо N_n. Тоді побудова послідовності елементів перестановок здійснюється за формулою:

$$P_n(N_n) = \bigcup_{i=n}^{1} P_{n-1}(N_n \setminus i)\, i. \qquad (3.2)$$

Отже, даний метод полягає в виконанні послідовності кроків:

1. Визначаємо послідовність чисел і впорядковуємо її за зростанням $p_1 \le p_2 \le \ldots \le p_n$. Присвоюємо $i := n$.

2. Фіксуємо елемент з індексом *i* в визначеній послідов-ності (на першому етапі це буде елемент – p_n). останні $(n-1)$-елементів генеруємо наступним чином:

а) шляхом переходу зліва направо знаходимо найлівішу позицію в якій $p_i > p_{i+1}$;

б) знайшовши її, шукаємо p_j, найменший елемент, розташований ліворуч і менший за нього;

в) здійснюємо транспозицію елементів p_j і p_i − отримаємо новий набір елементів, що визначає перестановку.

3. Зменшуємо на одиницю $i := i-1$, переходимо на крок 2.

Отже, шляхом послідовного генерування всіх елементів $p_{i_1}, p_{i_2}, \ldots, p_{i_{n-1}}$ при фіксованому останньому елементу p_{i_n} отримаємо всі перестановки. Таким чином, можемо виписати всі перестановки *n* елементів систематичним перебором всіх можливих для $p_n, p_{n-1}, \ldots, p_1$. Простий спосіб зробити це полягає в

тому, щоб почати з перестановки $(1,2,\ldots,n)$ і послідовно зміщувати на одну позицію всі $(n-1)$ елементів. Зміщення по циклу на один розряд елементів приводить до наступної породженої перестановки.

Зазначимо, що послідовності перестановок, одержані за допомогою вище зазначеного методу, можна інтерпретувати як граф G_n, вершини якого відповідають всім елементам множини перестановок $P(A)$.

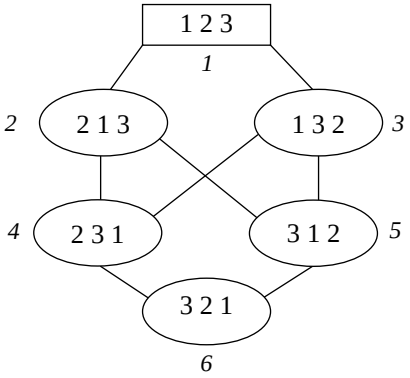


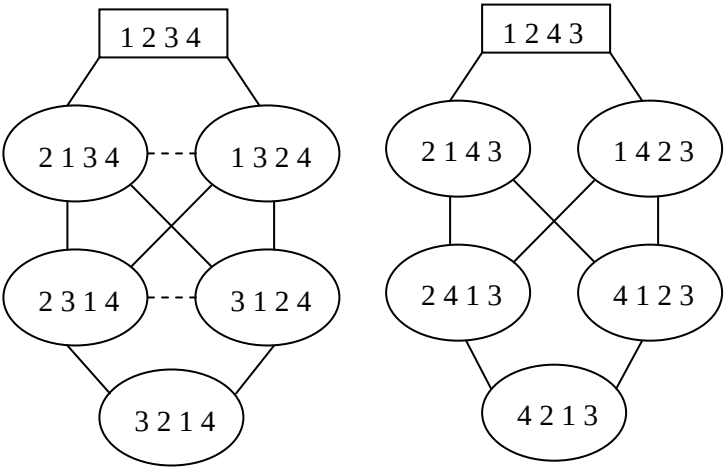
Рис. 3.4. Інтерпретація послідовності перестановок для $n=3$

У графі дві вершини, що відповідають перестановкам p_1 і p_2, сполучені ребром тоді і тільки тоді, коли p_2 утворюється з p_1 одноразовою транспозицією двох елементів. при фіксованому останньому.

Розглянемо тепер детально структуру відповідних графів для невеликих значень *n*. Для $n=3$ на рис. 3.4 зображено неорієнтований граф, що відображає утворення перестановки p_i з перестановки p_j рекурсивним методом.

Якщо кількість елементів множини, з якої утворюється перестановка, збільшується, то один з них фіксуємо, а останні переставляємо таким же методом. Для наглядної ілюстрації розглянемо метод генерування на прикладі $n=4$, представивши всі елементи перестановки як розкладання графа множини перестановок на підграфи, де виділемо деякий фіксований

елемент. Перший підграф представляє ті елементи конфігурації перестановок, в яких на останньому четвертому місці стоїть максимальний елемент – цифра 4. Підграф при $x_4 = 4$ представлений на рис. 3.5а якщо об'єднати перестановки, в яких на четвертому місці стоїть цифра 3, то одержимо підграф, представлений на рис. 3.5б. Аналогічно можна представити графи елементів конфігурації перестановок, де на останньому місці зафіксовані цифри 2 і 1 відповідно.



а) б)

Рис.3.5. Розкладання на підграфи послідовності перестановок для $n = 4$

Для представлення загального графа позначимо підграф, що представлений на рис. 3.5а – *A*. Таким чином, можна зобразити всі підграфи, які одержані з початкового *A* за допомогою однієї транспозиції. Слід відзначити, що у всіх підграфах однозначно визначається місце останнього елементу. І взагалі, всі підграфи є копією(проекцією) підграфа *A*, так що їх можна в тому ж порядку розташувати один під одним. Неважко помітити, що підграф *B* одержується з підграфа *A*, якщо в останньому у всіх перестановках зробити транспозицію чисел 4 і 3. Аналогічно підграф *C* отримаємо як проекцію підграфа *B*, якщо в останньому зробити транспозицію (3, 2), а підграф *D* з підграфа *C*

після транспозиції чисел 2, 1. враховуючи загальні поняття теорії графів та об'єднавши всі підграфи, одержимо загальний граф перестановок для $n=4$: $G_n(P)=A\cup B\cup C\cup D$ зображений на рис. 3.6.

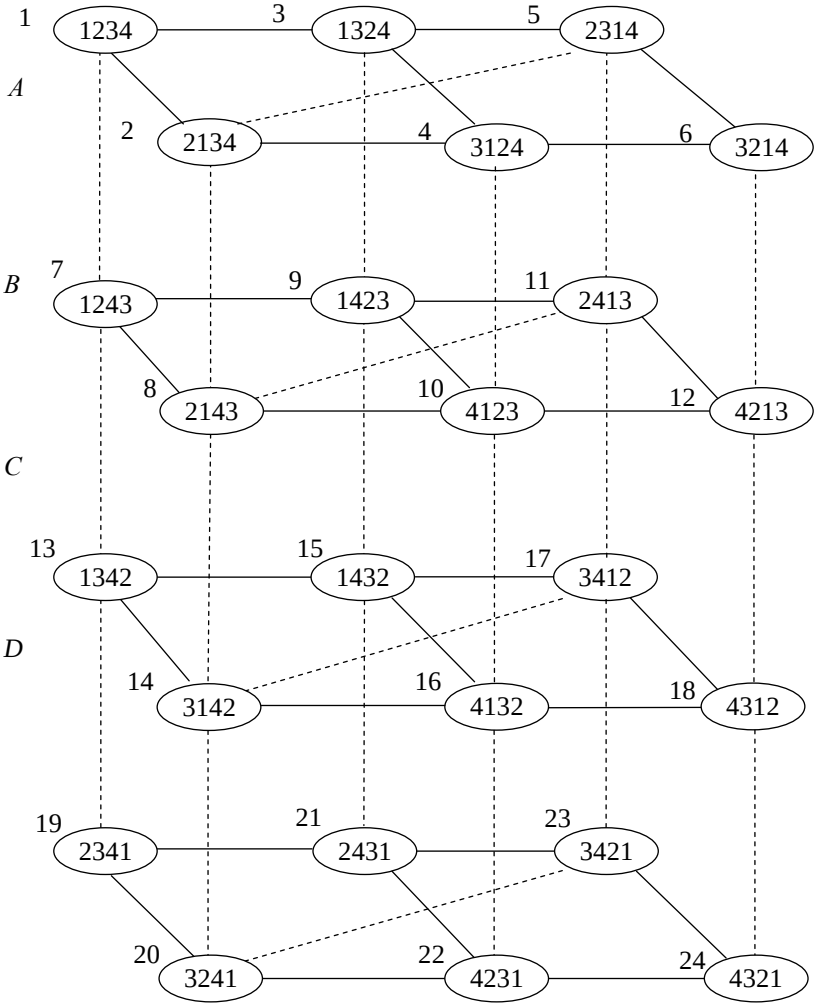


Рис. 3.6. Граф перестановок $G_n(P)$

Оскільки при цьому в підграфі $A(B,C)$ робиться одна транспозиція, то всі вершини копії з'єднуються ребром з відповідними вершинами оригіналу. В результаті одержуємо так званий ***граф перестановок,*** вершинами якого є елементи перестановки, і дві вершини в ньому суміжні, якщо відповідні перестановки відрізняються однією транспозицією елементів.

Побудову графа перестановок можна здійснити по індукції наступним чином. Спочатку для $n=2$ дві перестановки (12) та (21) як дві вершини з'єднаємо ребром. Переходимо до $n=3$. Для цього допишемо в кожній з двох вершин з правого боку число 3. На рис. 3.4 отримаємо вершини *1* і *2*. Тепер зробимо копію цього графа (цих вершин); в ньому виконаємо транспозицію $(3 \Leftrightarrow 2)$; З'єднуємо ребром відповідні вершини оригіналу та копії. На рис. 3.4 отримаємо вершини *3* і *5*. Нарешті робимо копію вершин *3* і *5*, виконуємо на них транспозицію $(2 \Leftrightarrow 1)$ і з'єднуємо ребром відповідні вершини першої та другої копій. Тим самим отримали граф перестановок $G_3(P)$. Далі зрозуміло: якщо побудовано граф перестановок $G_{n-1}(P)$, то дописуємо в кінці коду перестановок цього графа число *n*, отримуємо підграф A_1. Якщо отримано підграф $A_i\,(1 \le i < n)$, то робимо його копію, виконуємо на вершинах копії транспозицію $(n-i+1 \Leftrightarrow n-i)$, з'єднуємо ребром відповідні вершини оригіналу та копії. Тим самим отримуємо підграф A_{i+1}. Продовжуємо вказані дії до отримання підграфу A_n. Це і завершує побудову графа перестановок $G_n(P)$. На рис. 3.6 підграфи $A_i\,(i=1,2,3,4)$ відповіно позначені *A, B, C, D.*

Подібне представлення генерування комбінаторної конфігурації дає можливість розглядати екстремальні комбінаторні задачі на графі деякої множини. Таким чином надалі можемо оперувати з відомою структурою графа конфігурацій, а також надати йому орієнтацію в залежності від вигляду функції, яка оптимізується в екстремальних задачах.

Метод переміщення максимального елементу

Даний метод генерування елементів комбінаторної конфігурації передбачає вибір максимального елементу в множині

$P = (p_1, p_2, \ldots, p_n)$ з якої генеруються елементи комбінаторного об'єкту і його переміщення справа наліво. Переваги даного методу не є абсолютними по відношенню до інших графів перестановок, але в залежності від типу функції такий граф може сприяти побудові досконалішого алгоритму розв'язку поставленої задачі

Звичайно перестановку можна представити за допомогою таблиці з двома рядками, з яких перший містить всі елементи множини X в заданому порядку (як правило, в зростаючому), а другий – елементи $f(x)$, які розташовуються під відповідним елементом x.

Так як природа елементів множини X для подальших досліджень неістотна – приймемо для простоти $X = \{1, \ldots, n\}$, так само, як і в попередніх випадках, позначимо множину всіх перестановок цієї множини через P_n, а довільна перестановка $f \in P_n$ ототожнюватиметься з послідовністю $\langle a_1, \ldots, a_n \rangle$, де $a_i = f(i)$.

Визначимо суперпозицією перестановок f і g, під якою розумітимемо перестановку fg, що визначається таким чином:

$$fg(i) = f(g(i)).$$

Зазначимо, що для суперпозиції двох перестановок

$$f = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 3 & 2 & 1 & 4 \end{pmatrix} \quad g = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 5 & 3 & 1 & 4 \end{pmatrix},$$

достатньо змінити порядок стовпців в перестановці f так, щоб в першому рядку одержати послідовність, що є в другому рядку перестановки g, тоді другий рядок перестановки f дає суперпозицію fg.

Зазначимо, що кожна перестановка $f \in P_n$ однозначно визначає перестановку f^{-1}, таку що $ff^{-1} = f^{-1}f = e$, де перестановку

$$e = \begin{pmatrix} 1 & 2 & \dots & n \\ 1 & 2 & \dots & n \end{pmatrix}$$

називатимемо тотожною перестановкою. Очевидно, що $ef = fe = f$ для довільної перестановки $f \in S_n$. Називатимемо f^{-1} перестановкою, зворотною до *f*. Щоб її визначити, достатньо поміняти місцями рядки в записі перестановки *f*. Тоді для довільних перестановок $f, g, h \in S_n$ виконуються умови $(fg)h = f(gh)$.

Щоб відобразити цей факт, говоритимемо, що S_n утворює групу щодо операції суперпозиції. Цю групу називатимемо симетричною групою степеня *n*.

Довільну перестановку, що є циклом довжини 2, називатимемо транспозицією. Важливу роль в подальших міркуваннях виконуватимуть транспозиції сусідніх елементів, тобто транспозиції вигляду $[i, i+1]$.

Метод переміщення максимального елемента також зводиться до побудови підграфів меншого розміру, які відрізняються один від одного положенням елемента *n*. Тоді послідовність таких перестановок будується за допомогою наступної формули:

$$P_n(N_n) = \bigcup_{i=n}^{1} P_{n-1}(N_n \setminus n;\ n \to i\text{-}\textit{му місті}). \qquad (3.3)$$

Для більш наглядної ілюстрації розглянемо приклад для кількості елементів множини n=5, з якої формуються об'єкти перестановок. Нехай задано множину елементів перестановки $P = (p_1, p_2, \dots, p_n)$, які визначені числами $(1,2,3,4,5)$. На початковому етапі роботи методу вибирається початковий елемент, що є максимальним – в даному випадку 5, фіксується його позиція на останньому місці справа. Розглянемо граф формування підпослідовності перестановок при фіксованому останньому елементі $x_5 = 5$.

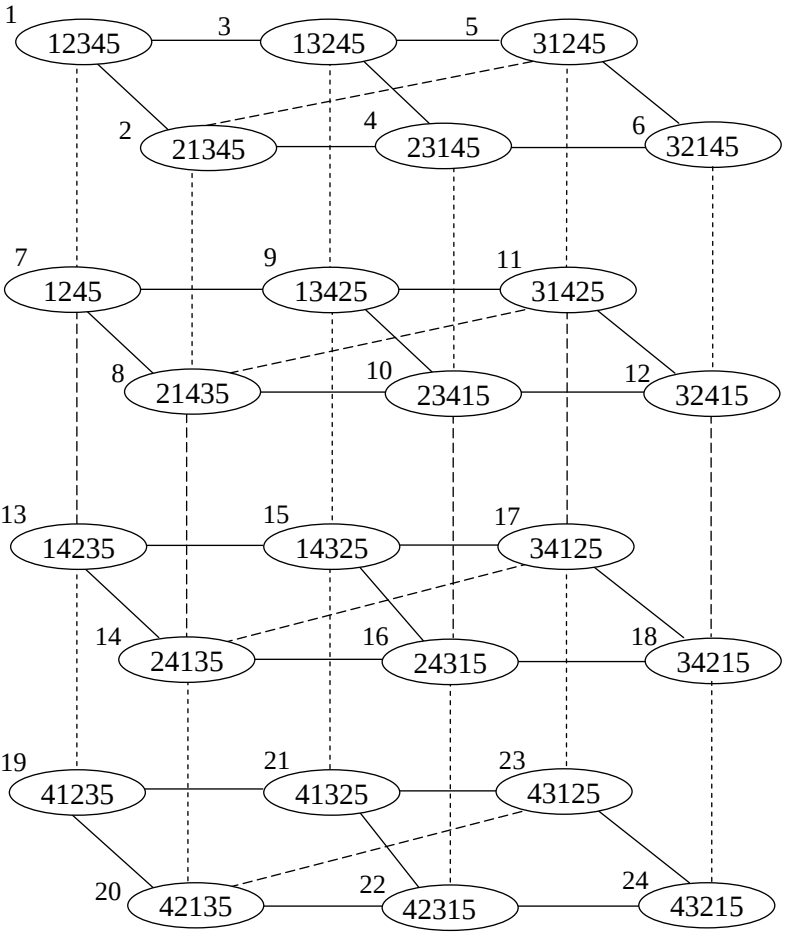


Рис. 3.7. Підпослідовність перестановок при фіксованому значенні $x_5 = 5$

Далі максимальний елемент переміщується на одну позицію вліво, тобто займає четверте місце. Тоді наступну підпослідовність перестановок можна представити у вигляді графа на рис. 3.8.

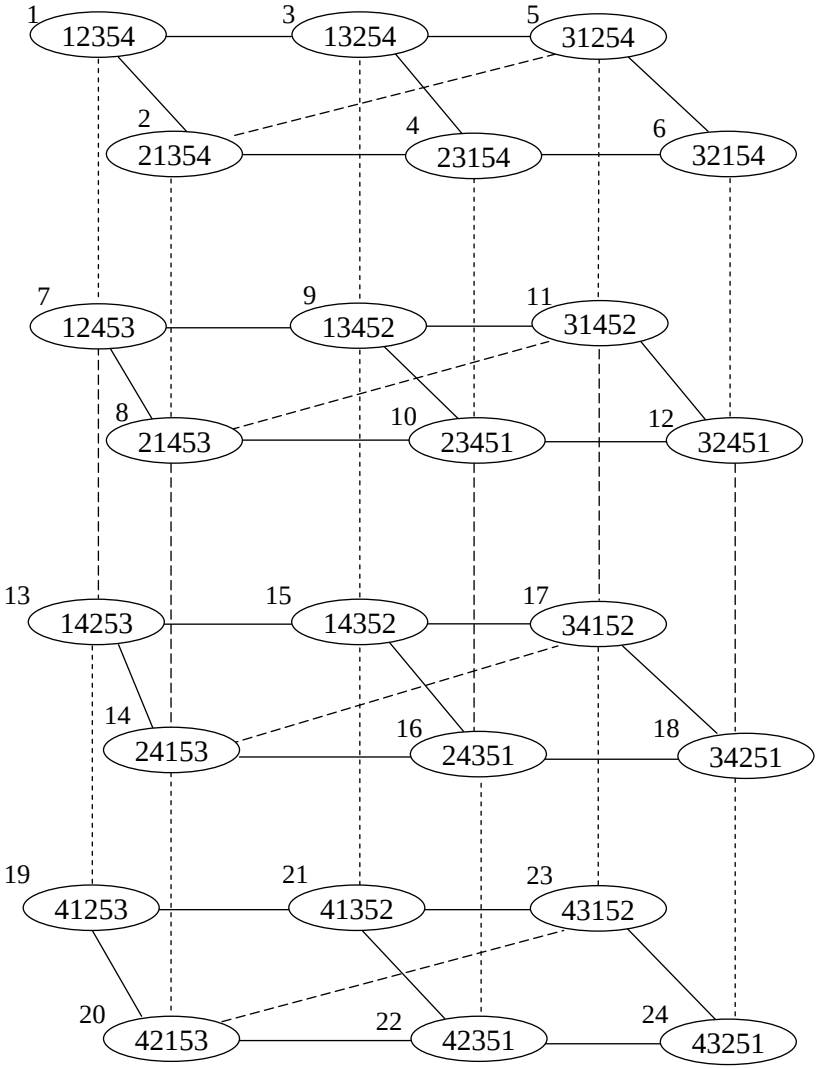


Рис. 3.8. Підпослідовність перестановок
при фіксованому значенні $x_4 = 5$

Наступна позиція, яку займає максимальний елемент визначається $x_3 = 5$. Тоді підпослідовність зображується у вигляді наступного графа.

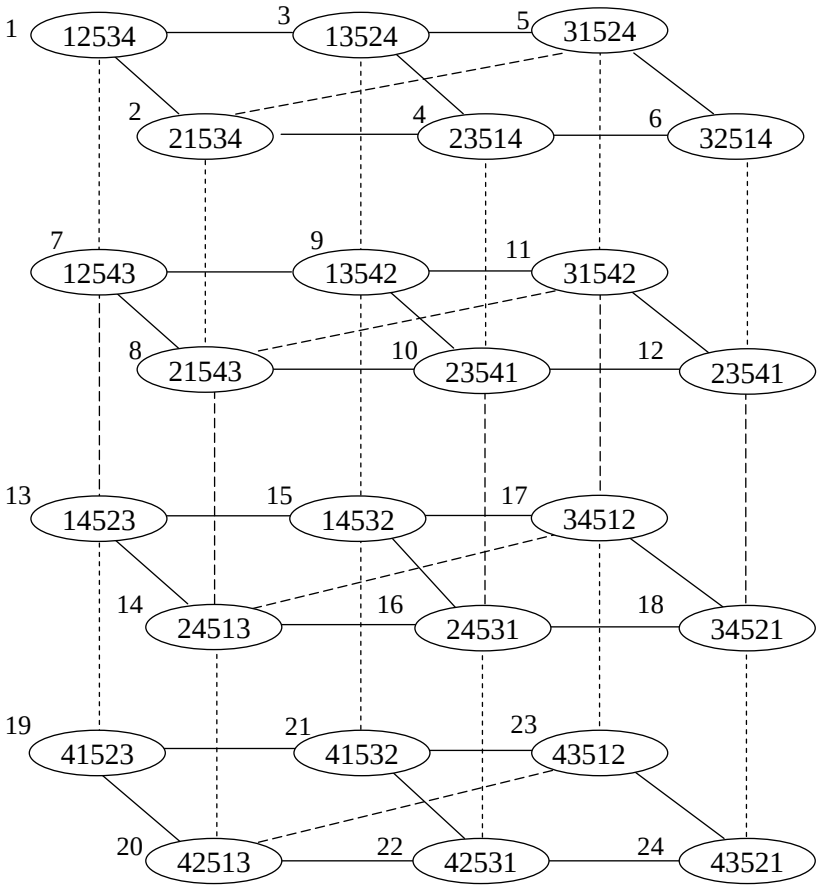


Рис. 3.9. Підпослідовність перестановок при фіксованому значенні $x_3 = 5$

Відповідно далі зображаються графи, на яких представлені підпослідовності перестановок, що утворюються при позиції максимального елементу $x_2 = 5,\ x_1 = 5$.

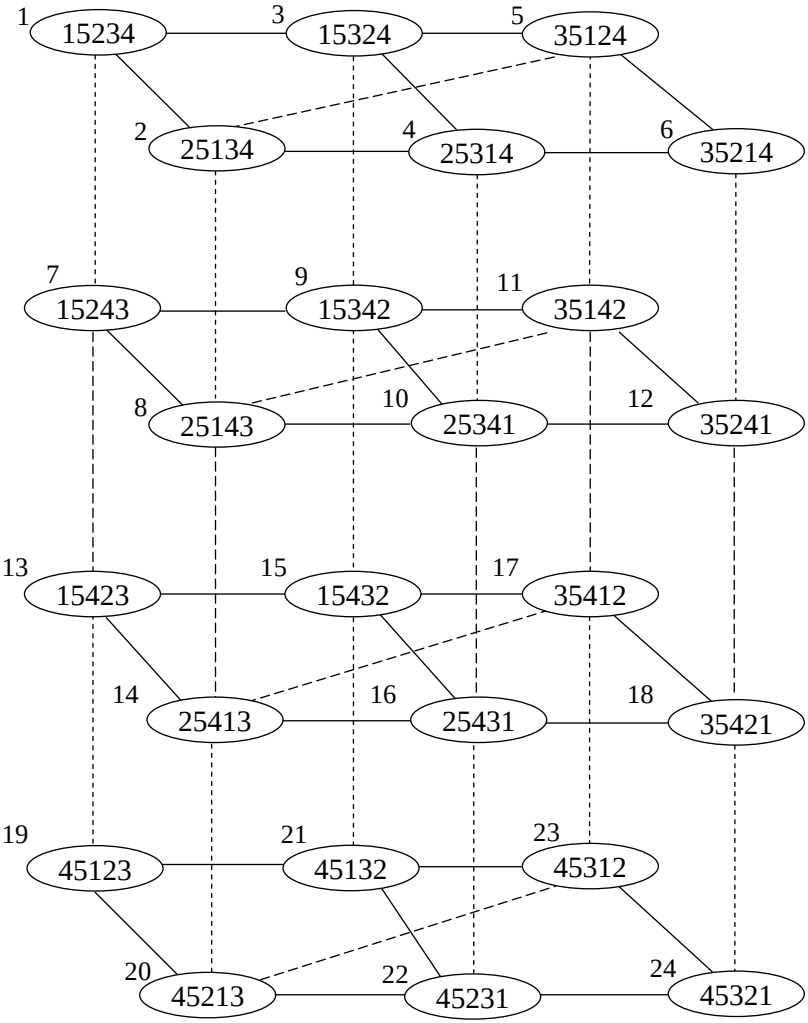


Рис. 3.10. Підпослідовність перестановок
при фіксованому значенні $x_2 = 5$

Далі максимальний елемент переміщується на перше місце і розглядається підграф, в якому вершини визначаються точками перша координата яких $x_1 = 5$.

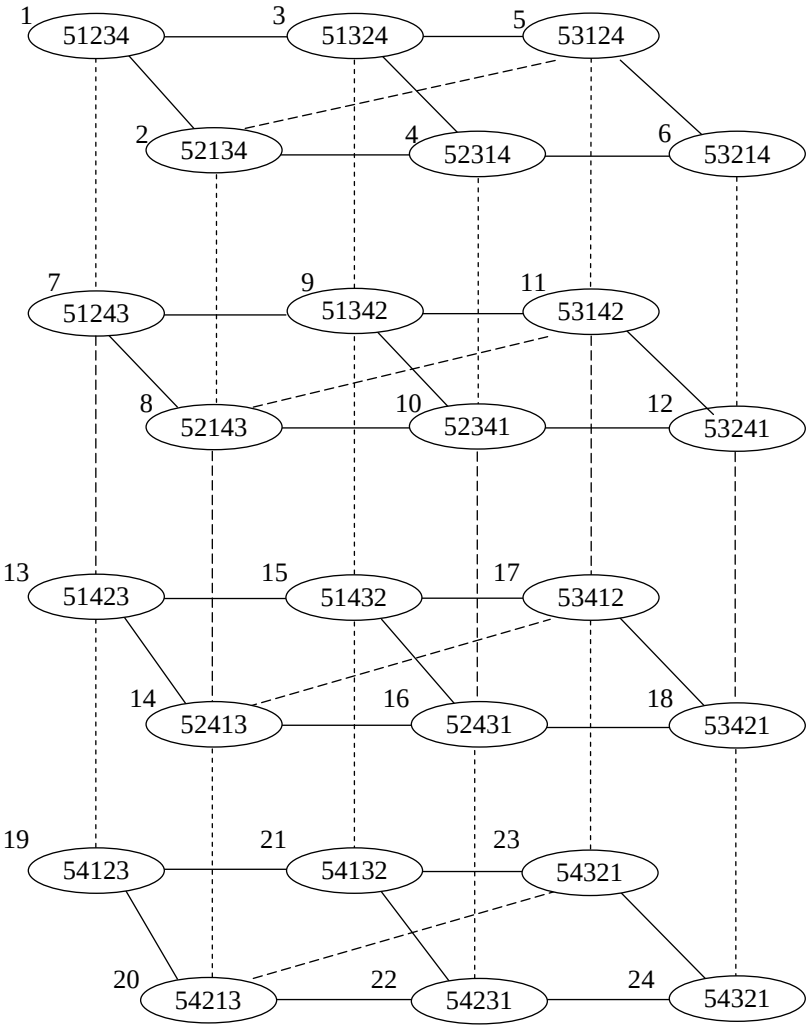


Рис. 3.11. Підпослідовність перестановок
при фіксованому значенні $x_1 = 5$

Розподіл інших елементів координат вершин залишається сталим, як і в інших підграфах, де максимальний елемент знаходиться на іншому місці.

Отже, на основі вище описаного прикладу, можна привести загальний опис рекурентного алгоритму генерування перестановок згідно методу переміщення максимального елементу.

Спочатку для $n=2$ дві перестановки (12) та (21) як дві вершини з'єднаємо ребром. Переходимо до $n=3$. Для цього зробимо три копії цього ребра. В першій копії допишемо число 3 на третє місце. На рис.3.4 отримаємо вершини *1* і *2*. В другій копії допишемо число 3 на друге місце, що відповідає вершинам *3* і *4* на рис. 3.4., і з'єднаємо відповідні вершини першої та другої копій. Тепер допишемо число 3 в третій копії на перше місце, що відповідає вершинам *5* і *6* на рис. 3.4, і з'єднаємо відповідні вершини другої та третьої копій.

Тим самим отримали граф перестановок $G_3(P)$. Далі зрозуміло: якщо побудовано граф перестановок $G_{n-1}(P)$, то робимо *n* копій цього графа, в *i*-й копії $(1 \le i \le n)$ на *i*-му місці в усіх перестановках дописуємо число *n*. Далі з'єднаємо відповідні вершини *i*-ї та $(i+1)$-ї копій. Тим самим побудовано граф перестановок $G_n(P)$.

Слід зазначити, що особливістю нових методів генерування комбінаторної конфігурації у вигляді перестановок незалежно від способу генерування є те, що кожний з цих методів дає можливість представити граф, який можна побудувати за деяким способом генерування. В графі вершини розміщені згідно деякої визначеної властивості, тому можна зразу визначити крайні вершини графа, далі граф можна розкласти на підграфи. Для розв'язування екстремальних задач є важливим представлення всієї структури графа з допомогою графів меншої розмірності, або більш простої структури.

Отже, нові методи генерування самі по собі не є важливими, а головним є те, що вони дають можливість користуватися графом, а не зберігати всі об'єкти заданої конфігурації. Надалі є важливим орієнтувати граф та визначити інцидентність вершин та дуг.

Оскільки екстремальні задачі оптимізації розглядаються на різних комбінаторних конфігураціях, то далі розглянемо основні властивості та проаналізуємо методи генерування інших комбінаторних конфігурацій, а також побудову нових методів генерування об'єктів конфігурації і представлення їх в вигляді графів.

3.2.2. Генерування розміщень

Генерація розміщень в загальному випадку має елементарний характер і нараховує безліч методів. Враховуючи те, що для перестановок можна побудувати граф, за допомогою якого можна представити процедуру генерування комбінаторних об'єктів перестановки, розглянемо представлення генерування множини розміщень довжини m з елементами з множини $X = \{1, 2, \ldots, n\}$ у вигляді графа, який позначимо $G_m(A_n)$. При цьому необхідно визначити, які вершини графа рахувати суміжними, тобто з'єднані ребром. Якщо у перестановках бралися до уваги коди суміжних перестановок, які відрізнялися лише однією транспозицією, то у розміщеннях це неможливо, так як вони можуть взагалі не мати спільних елементів. Будемо вважати суміжними два розміщення, які відрізняються або однією транспозицією, або лише одним елементом на одному і тому ж місці. Наведемо приклад генерування розміщень другого типу (за формулою 3.1) для послідовності довжини 3 з елементами з множини $X = \{1, 2, 3, 4\}$. На графі можна представити дані об'єкти в вигляді рис. 3.12.

Дану побудову можна представити також і рекурентним методом. Отже, маємо два числа $n > m \geq 1$ і треба побудувати граф розміщень з $A_n(m) = [n]_m$ вершин. На початку розглянемо випадок $n = m + 1$ (на рис. 3.12). Будуємо граф перестановок для $G_m(P)$, для чого можемо скористатися одним з представлених вище методів. На рис. 3.12 це $G_3(P)$, (вершини 19–24). Далі робимо $n-1$ копій цього графа. В першій копії (вершини 13–18) робимо скрізь заміну $n-1$ на n (3 на 4) та з'єднуємо ребрами (пунктирними) відповідні вершини (13, 19), (14, 20) і т. д. Далі по індукції в i-й копії $(n-1 \geq i \geq 1)$ всюди робимо заміну елементів $n-i$ на $n-i+1$, потім з'єднуємо ребрами відповідні вершини i-ї та $(i-1)$-ї копій. Цим і завершується побудова графа розміщень $G_m(A_{m+1})$. Як бачимо, суміжність вершин в цьому графі має двоякий вигляд – в $G_3(P)$ та його копіях, як і в

перестановках, завдяки одній транспозиції, а вершин з різних копій – відмінністю тих двох елементів, які замінюються на одному місці. Оскільки для $n = m+1$ число розміщень дорівнює $m(m-1)(m-2)2 = m!$, то очевидно, що при цьому граф розміщень співпадає з графом перестановок. Загальна схема рекурентної побудови графа розміщень для довільних *n* та *m* дещо відрізняється.

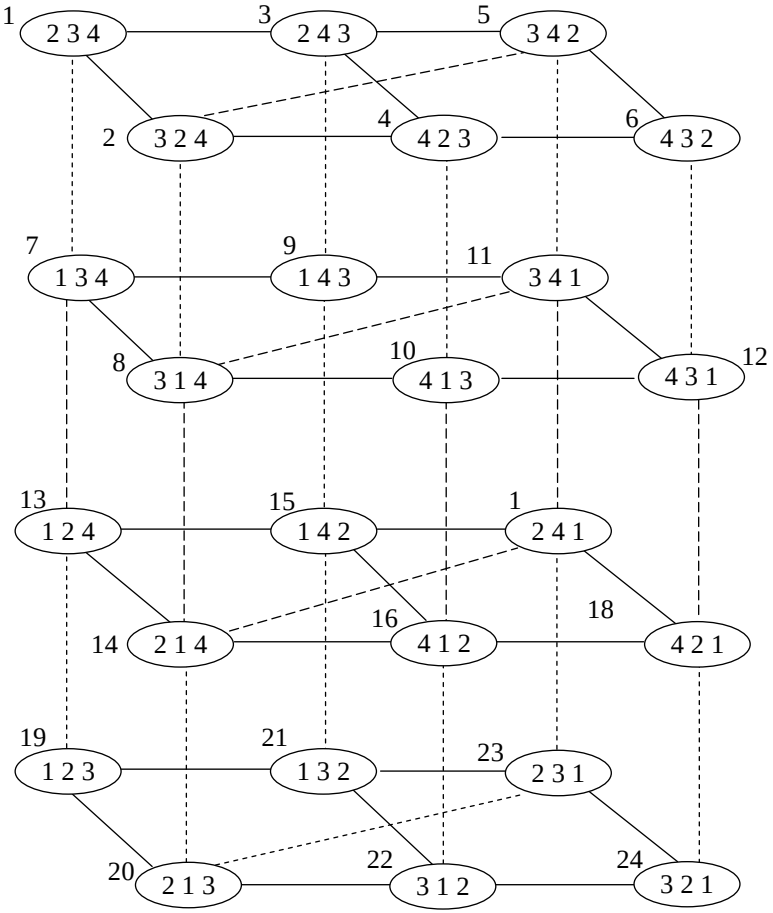


Рис. 3.12. Граф множини розміщень для $n = 4$ та $m = 3$

Почнемо з побудови графа $G_1(A_n)$. Це ізольовані вершини з номерами $1, 2, \ldots, n-1, n$.

Побудова графа $G_m(A_n)$ ведеться по наступній індукції: спочатку будуємо граф $G_1(A_{n-m+1})$. Для цього припускається, що побудовано граф розміщень $G_i(A_{n-m+i})$ $(m \geq i \geq 1)$, де довжина коду розміщень дорівнює i. Далі робимо i +1 копію графа $G_i(A_{n-m+i})$. В j-й копії $(i+1 \geq j \geq 1)$ зправа в кожному розміщенні дописуємо число j. При цьому, якщо таке число в роміщенні є зліва, то воно замінюється числом i+1. З'єднуємо ребрами відповідні вершини j-ї та $(j-1)$-ї копій $(j>1)$. Якщо такі дії виконані для кожного j від 1 до $i+1$, то тим самим побудовано граф $G_{i+1}(A_{n-m+i+1})$.

Коли довжина коду розміщень стане рівною m, то це буде означати, що отримано граф розміщень $G_m(A_n)$. Зазначимо, що відомий граф $G_1(A_3) = 1, 2, 3$.

Нижче наведено таблицю, яка ілюструє дію цього алгоритму для побудови $G_3(A_5)$.

Таблиця 3.1

Побудова підграфів

$G_2(A_4)$	$G_3(A_5)$				
4,1	4,5,1	4,1,2	4,1,3	5,1,4	4,1,5
2,1	2,5,1	5,1,2	2,1,3	2,1,4	2,1,5
3,1	3,5,1	3,1,2	3,1,3	3,1,4	3,1,5
1,2	5,2,1	1,5,2	1,2,3	1,2,4	1,2,5
4,2	4,2,1	4,5,2	4,2,3	5,2,4	4,2,5
3,2	3,2,1	3,5,2	5,2,3	3,2,4	3,2,5
1,3	5,3,1	1,3,2	1,5,3	1,3,4	1,3,5
2,3	2,3,1	5,3,2	2,5,3	2,3,4	2,3,5
4,3	4,3,1	4,3,2	4,5,3	5,3,4	4,3,5
1,4	5,4,1	1,4,2	1,4,3	1,4,4	1,4,5
2,4	2,4,1	5,4,2	2,4,3	2,4,4	2,4,5
3,4	3,4,1	3,4,2	5,4,3	3,4,4	3,4,5

3.2.3. Генерування розбиття множини та числа

Під розбиттям *n*-елементної множини *X* на *k* блоків будемо розуміти будь-яке сімейство $\pi = \{B_1 \cup B_2 \cup \ldots \cup B_k\}$, що $B_1 \cup B_2 \cup \ldots \cup B_k = X$, $B_i \cap B_j = \varnothing$ для $1 \le i < j \le k$ та $B_i \ne \varnothing$ для $1 \le i \le k$. Підмножини $B_1, B_2, \ldots, B_k$ будемо називати блоками сімейства π. Множина всіх розбиттів множини *X* на *k* блоків будемо позначати через $\Pi(X)$. Очевидно, що $\Pi(X) = \Pi_1(X) \cup \ldots \cup \Pi_n(X)$ (більше того, $\{\Pi_1(X), \ldots, \Pi_n(X)\}$ є розбиттям множини $\Pi(X)$).

Опишемо алгоритм генерування всіх розбиттів множини у вигляді графа. Даний алгоритм легко пояснити, сформулювавши його наступним чином: відмітимо спочатку, що кожне розбиття π множини $\{1,\ldots,n\}$ однозначно визначає розбиття π_{n-1} множини $\{1,\ldots,n-1\}$, що виникає із π після видалення елемента *n* із відповідного блоку (і видалення утворюваного пустого блоку, якщо елемент *n* утворював одноелементний блок). Навпаки, якщо дано розбиття $\sigma = \{B_1, \ldots, B_k\}$ множини $\{1,\ldots,n-1\}$, легко знайти всі розбиття π множини $\{1,\ldots,n\}$ такі, що $\pi_{n-1} = \sigma$, тобто наступні розбиття:

$$
\begin{aligned}
&B_1 \cup \{n\}, B_2, \ldots, B_k \\
&B_1 B_2, \cup \{n\}, \ldots, B_k \\
&\ldots\ldots\ldots\ldots\ldots\ldots \\
&B_1 B_2, \ldots, B_k \cup \{n\} \\
&B_1 B_2, \ldots, B_k \{n\}
\end{aligned}
\qquad (3.3)
$$

Рис. 3.13. Послідовність розбиттів

Тоді рекурентний метод генерування розбиття множини полягає в наступному: якщо отримано список L_{n-1} всіх розбиттів множини $\{1,\ldots,n-1\}$, то список L_n всіх розбиттів множини $\{1,\ldots,n\}$ будемо одержувати, замінюючи кожне розбиття σ в

списку L_{n-1} на відповідну йому послідовність (3.3), згідно рис. 3.14. Зазначимо, що при цьому можна гарантувати, що розбиття, які йдуть один за одним, мало відрізнятиметься одне від одного, точніше кажучи, що кожне наступне розбиття в списку утворюється з попереднього за допомогою видалення деякого елементу з деякого блоку і додавання його в інший блок або створення з нього одноелементного блоку.

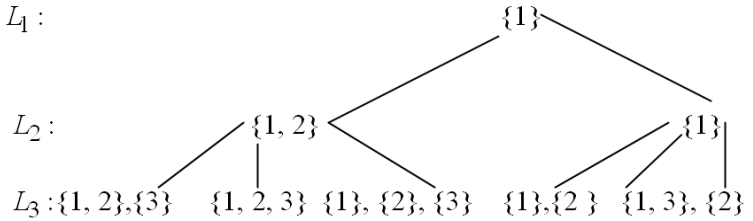


Рис. 3.14. Побудова списків L_1 L_2

Дійсно, послідовні розбиття послідовності (3.3) відповідають цій умові. Якщо порядок послідовності зробити зворотним $\{3, 3\}$ для кожного другого розбиття списку L_{n-1}, то елемент n буде переміщуватися почергово вперед і назад, і розбиття «на стику» послідовностей, утворених із сусідніх розбиттів списку L_{n-1}, будуть мало відрізнятися одне від одного (при умові, що сусідні розбиття списку L_{n-1} мало відрізняються одне від одного). На рис. 3.14 показано побудову списку L_n для множини елементів 1, 2, 3.

Цей спосіб генерування будує спочатку розбиття $\{1,\ldots,n\}$ потім здійснюється переміщення «активного» елементу j у сусідній блок − попередній або подальший (у останньому випадку може виникнути необхідність створення нового блоку елементів, а потім визначення активного елементу в новоутвореному розбитті. З описаної побудови випливає, що даний елемент переміщається тільки тоді, коли всі елементи, що більші нього, досягають свого крайнього лівого або правого положення; точніше, активний елемент j^* є таким найменшим елементом.

Генерування розбиття числа зручно задавати розбиттям $\{p_1, p_2, \ldots, p_k\}$ числа n із компонентами, розташованими в деякому порядку, наприклад $p_1 \le p_2 \le \ldots \le p_k$.

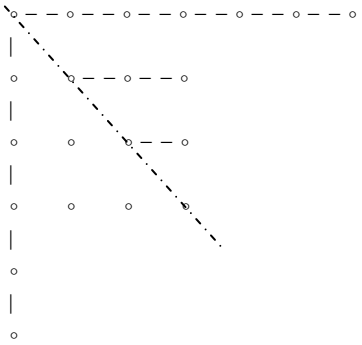
Розбиття n із l компонентами можна породжувати в зростаючому лексикографічному порядку, починаючи із $p_1 = p_2 = \ldots = p_{l-1} = 1, \quad p_l = n - l + 1$ продовжуючи процес таким чином. Для отримання наступного розбиття з поточного проглядаємо елементи справа наліво, зупиняючись на найправішому p_i, такому, що $p_l - p_i \ge 2$. Замінюємо потім p_j на $p_j + 1$ для $j = i,\ i+1, \ldots, l-1$ і після цього замінюваний p_l на $n - \sum_{j=1}^{i-1} p_j$.

Іншим ефективним елементарним інструментом вивчення розбиття служить їх графічне представлення. Кожному розбиттю числа ставиться у відповідність його графічне представлення G_λ в вигляді точкового графу. Точковий граф розбиття $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n)$ представляє собою множину цілочислових точок (i, j) на площині, для яких виконується наступна умова

$$(i, j) \in G_\lambda \Leftrightarrow 0 \ge i \ge -n+1,\ 0 \le j \le \lambda_{i+1} - 1,$$

де i – вертикальна вісь, j – горизонтальна вісь.

Точковий граф представляє собою наступну графічну структуру, де частини розбиття розміщуються як правило зверху вниз по спаданню.



3.2.4. Генерування сполучень

Розглянемо методи генерування комбінаторної конфігурації сполучень, тобто генерування *k*-елементних підмножин.

Розглянемо множину $A' = \{1,2,...,n\}$. Сполучення без повторень з *n* елементів по *r* – це *r*-елементна підмножина множини A'. Оскільки порядок запису елементів множини неістотний, то домовимося записувати елементи в кожному сполученні у порядку зростання. Отже, сполучення $\{a_1, a_2, \ldots, a_r\}$, розглядатимемо як рядок чисел $a_1, a_2, \ldots, a_r$, причому $a_1 < a_2 < \ldots < a_r$.

Кількість усіх сполучень без повторень з *n* елементів по *r* позначимо як C_n^r, де *r* і *n* – невід'ємні цілі числа, причому $r \le n$. Кількість усіх сполучень із повтореннями з *n* елементів по *r* позначимо як H_n^r або $H(n,r)$, де *r* і *n* – будь-які невід'ємні цілі числа. Числа C_n^r називають біноміальними коефіцієнтами.

Як і для перестановок, за даним сполученням можна знайти наступне відповідно до лексикографічного порядку. Значення останнього числа в рядку – найбільш можливе, якщо воно дорівнює $n = n - r + r$. Якщо останнє число – найбільш можливе, то передостаннє – найбільш можливе, якщо воно дорівнює $n - r + (r-1)$ або $n - r + i$, де $i = r - 1$ – позиція цього числа. Загалом, значення кожного *i*-го числа найбільш можливе, якщо числа праворуч від нього – найбільші можливі, і це значення дорівнює $n - r + i$. Отже, проглядаємо рядок справа наліво й визначаємо, чи дорівнює значення *i*-го елемента $n - r + i$ (це максимальне значення, яке може бути в *i*-й позиції). Перше значення, яке не задовольняє цю умову, можна збільшити. Нехай, наприклад, це значення дорівнює *m* і займає *j*-ту позицію. Збільшуємо *m* на 1, а значення кожного елемента, що стоїть після *j*-го, дорівнює значенню попереднього елемента плюс 1. Тепер можемо сформулювати потрібний алгоритм.

Алгоритм побудови лексикографічно наступного сполучення полягає у виконанні наступних кроків:

Крок 1. Знайти в рядку $a_1, a_2, \ldots, a_r$ перший справа елемент a_i такий, що $a_i \ne n - r + i$.

Крок 2. Для знайденого елемента виконати присвоювання $a_i := a_i + 1$.

Крок 3. Для $j = i+1, i+2, \ldots, r$, виконати $a_{j^{\cdot}} = a_i + j - i$ (або, що те саме, $\left(a_{j^{\cdot}} = a_{j-1} + 1\right)$).

Даний алгоритм дає можливість будувати наступне в лексикографічному порядку сполучення. Рядок чисел, яким подано лексикографічно наступне сполучення, відрізняється від рядка, що зображає дане сполучення, з позиції *i*, бо в даному сполученні в позиціях $i+1, i+2, \ldots, r$ є максимально можливі числа. Отже, $a_i + 1$ – найменше можливе число, яке можна записати в позицію *i*, якщо хочемо отримати сполучення, більше від даного. Тоді $a_i + 2, \ldots, a_i + r - i + 1$ – найменш можливі числа, які можна записати в позиціях від $i+1$ до *r*.

На основі вище описаного алгоритму можна побудувати послідовність розміщень з *n* елементів по *r*.

Тоді розглядатимемо вище сформульовану задачу лише для множини $A' = \{1, 2, \ldots, n\}$. Один із можливих способів її розв'язання такий. Використаємо алгоритм генерування лексикографічно наступного сполучення для побудови *r*-елементних сполучень *n*-елементної множини A'. Після кожної стадії, коли побудовано чергове *r*-сполучення застосуємо $r! - 1$ разів алгоритм побудови перестановки за умови $n = r$ для побудови всіх перестановок елементів цього сполучення як *r*-елементної множини.

Розглянемо два відомі алгоритми генерування всіх *k*-елементних підмножин *n*-елементної підмножини *X* [148]. Без обмеження загальності можна прийняти $X = \{1, \ldots, n\}$. Тоді кожній *k*-елементній підмножині взаємно однозначно відповідає зростаюча послідовність довжини *k* з елементами з *X:* наприклад, підмножині $\{3, 5, 1\}$ відповідає послідовність $\langle 1, 3, 5 \rangle$.

Розглянемо алгоритм, за допомогою якого генеруються всі такі послідовності в лексикографічному порядку. Достатньо з цією метою зазначити, що при лексикографічному порядку послідовністю, що безпосередньо є наступною за послідовністю $\langle a_1, \ldots, a_n \rangle$, являється послідовність, що визначається наступним чином

$$\langle b_1, \ldots, b_k \rangle = \langle a_1, \ldots, a_{p-1}, a_p + 1, a_p + 2, \ldots, a_p + k - p + 1 \rangle,$$

де $p = max\{i : a_i < n - k + 1\}$.

Більш того, послідовністю, безпосередньо слідуючою за $\langle b_1, \ldots, b_k \rangle$, являється

$$\langle c_1, \ldots, c_k \rangle = \langle b_1, \ldots, b_{p'-1}, b_{p'} + 1, b_{p'} + 2, \ldots, b_{p'} + k - p' + 1 \rangle,$$

де $p' = \begin{cases} p - 1, & якщо \quad b_k = n, \\ k, & якщо \quad b_k < n \end{cases}$

(будемо вважати, що послідовності $\langle a_1, \ldots, a_k \rangle$ і $\langle b_1, \ldots, b_k \rangle$ відмінні від $\langle n - k + 1, \ldots, n \rangle$ останньої послідовності в нашому порядку). Послідовність всіх 4-елементних підмножин множини $\{1, \ldots, 6\}$, отримана за допомогою лексикографічного порядку, можна представити наступним чином рис 3.15.

1 2 3 4
1 2 3 5
1 2 3 6
1 2 4 5
1 2 4 6
1 2 5 6
1 2 4 5
1 3 4 6
1 3 5 6
1 4 5 6
2 3 4 5
2 3 4 6
2 3 5 6
2 4 5 6
3 4 5 6

Рис. 3.15. Послідовність 4-елементних підмножин множини $\{1, \ldots, 6\}$, побудована за допомогою лексикографічного порядку

Другий алгоритм, генерує всі *k*-елементні підмножини так, що кожна наступна підмножина утворюється з попередньої з видаленням одного елемента і додаванням іншого. Цей алгоритм представимо у рекурсивній формі.

Позначимо з цією метою через $G(n,k)$ список, який має всі *k*-елементні підмножини множини $\{1,\ldots,n\}$, в якій першою підмножиною є $\{1,\ldots,k\}$, останньою – $\{1,2,\ldots,k-1,n\}$ і кожна наступна підмножина утворюється з попередньої з видаленням деякого елемента і добавлянням другого. Зазначимо, що якщо $G(n-1,k)$ і $G(n-1,k-1)$ вже побудовані, то $G(n,k)$ можна визначити наступним чином:

$$G(n,k)=G(n-1,k-1),G^*(n-1,k-1)\cup\{n\},$$

де $G^*(n-1,k-1)\cup\{n\}$ означає список, утворений з $G(n-1,k-1)$ зміною порядку елементів списку на зворотний та наступним добавляння елемента *n* до кожної множини. Дійсно $G(n-1,k)$ містить всі *k* елементні підмножини множини $\{1,\ldots,n\}$, які не містять *n*, а $G^*(n-1,k-1)\cup\{n\}$ всі *k* елементні підмножини, які містять n, причому останньою підмножиною в списку $G(n-1,k)$ являється $\{1,2,\ldots,k-1,n-1\}$, а першою підмножиною в списку $G^*(n-1,k-1)\cup\{n\}$ являється $\{1,2,\ldots,k-1,n\}$. На рисунку 3.16 показаний процес побудови списку $G(4,2)$.

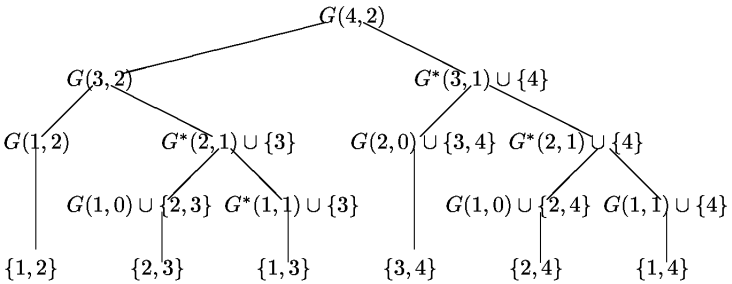


Рис. 3. 16. Побудова списку $G(4,2)$

Списку $G(n,k)$ можемо так як і у випадку генерування всіх підмножин, поставити в відповідність деякий гамільтонів шлях в графі, вершини якого відповідають двоелементним підмножинам множини $\{1,2,3,4\}$, причому вершини, які відповідають підмножинам *A* і *B*, з'єднані ребром тоді і тільки тоді, коли $|A \cap B| = k-1=1$. Тоді це можна проілюструвати на рис 3.17:

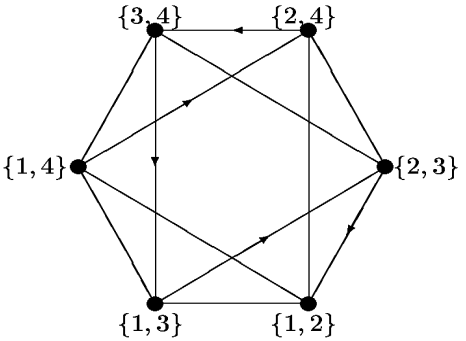


Рис. 3.17. Гамільтонів шлях в графі, що відповідає усім двоелементним підмножинам множини $\{1,2,3,4\}$

Даний список та граф не дає можливості побудувати упорядкування по значенням цільової функції.

Далі розглянемо нові методи генерування комбінаторної конфігурації сполучень, які дають можливість в залежності від складності задачі, представити елементи комбінаторної конфігурації у вигляді графа.

Даний метод генерування полягає у виборі елементів з заданої упорядкованої множини за зростанням, тобто вибираються елементи для прикладу по два перший, другий; перший третій; перший четвертий, і т.д. таким чином будується верхній підграф загального графу послідовності, далі – другий, третій; другий, четвертий, і т. д.

Враховуючи, що кількість усіх сполучень без повторень з *n* елементів по *r* дорівнює C_n^r, де *r* і *n* – невід'ємні цілі числа, причому $r \le n$, то розглянемо приклади побудови для наступних множин.

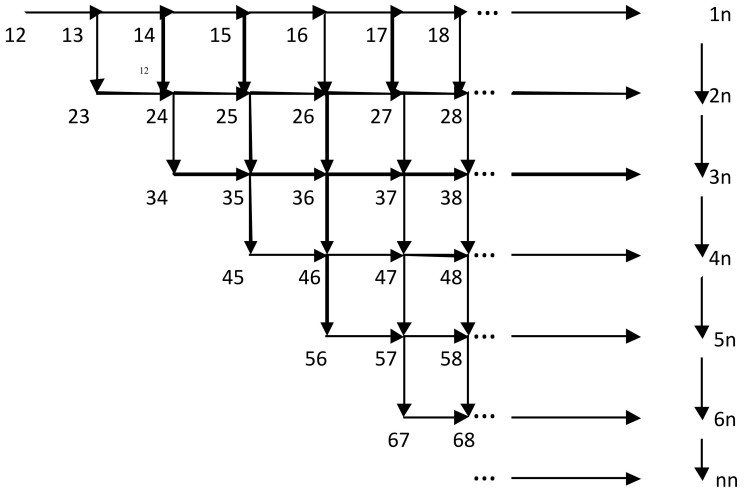


Рис. 3.18. Граф сполучень з *n* по 2

Розглянемо сполучення з 6 по 3, де $C_6^3 = 20$. Формування елементів сполучення здійснюється наступним чином: вибираються з впорядкованої за зростанням множини з шести заданих елементів 1, 2, 3, 4, 5, 6 перші три 1, 2, 3 і для побудови першого підграфу останній елемент замінюється наступними з заданої множини, маємо: 1, 2, 4; 1, 2, 5; 1, 2, 6. Тоді другий підграф утворюється шляхом заміни в кожному з попередніх елементів крім першого другої компоненти на наступний елемент з заданої множини, маємо 1, 3, 4; 1, 3, 5; 1, 3, 6.

Граф на рис. 3.19 можна інтерпретувати як простий тип графів – дерево, оскільки в ньому будь-які дві вершини з'єднані простим ланцюгом.

Дане дерево можна представити у вигляді розкладу на піддерева, тоді кожне піддерево формується з вершин в яких елемент на останньому місці фіксується і є вибраним з початкової множини $a_1, a_2, \ldots, a_n$ як максимальний, а останні перебираються рекурсивним методом.

Слід зазначити, що дане дерево є орієнтованим, в якому визначений корінь – вершина (4 5 6), яка є лексикографічно більша останніх.

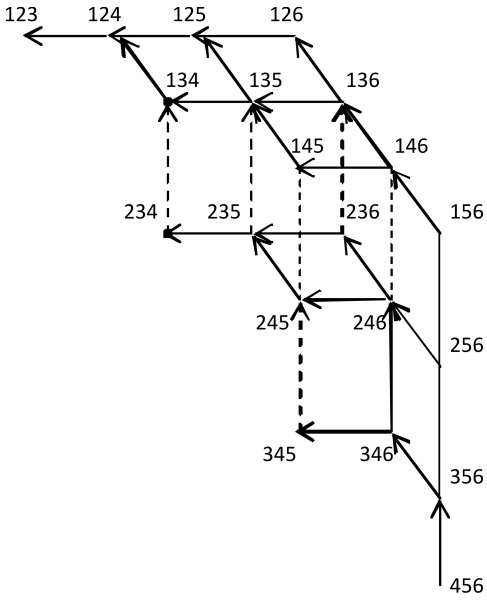


Рис. 3.19. Граф сполучень з 6 по 3, де $C_6^3 = 20$

Тепер можемо сформулювати потрібний алгоритм.

Алгоритм побудови наступного сполучення полягає в виконанні наступних кроків:

Крок 1. Знайти в рядку $a_1, a_2, \ldots, a_r$ перший справа елемент a_i такий, що $a_i \neq n - r + i$.

Крок 2. Для знайденого елемента виконати присвоювання $a_i := a_i + 1$.

Крок 3. Для $j = i+1, i+2, \ldots, r$, виконати $a_j := a_i + j - i$ (або, що те саме, $\left(a_j := a_{j-1} + 1\right)$.

Даний алгоритм дає можливість будувати наступне в порядку сполучення. Рядок чисел, яким подано лексикографічно наступне сполучення, відрізняється від рядка, що зображає дане сполучення, з позиції i, бо в даному сполученні в позиціях $i+1, i+2, \ldots, r$ є максимально можливі числа. Отже, $a_i + 1$ – найменше можливе число, яке можна записати в позицію i, якщо хочемо отримати сполучення, більше від даного. Тоді

$a_i + 2, \ldots, a_i + r - i + 1$ – найменш можливі числа, які можна записати в позиціях від $i+1$ до r.

Як уже було зазначено, відносно інших комбінаторних конфігурацій важливість запропонованого методу генерування комбінаторної конфігурації сполучень в тому, що одержуємо граф, в даному випадку дерево, який може бути використаний для розробки ефективного методу для розв'язування екстремальних задач, оскільки граф відображає всі елементи сполучень.

3.3. Загальна постановка методу направленого структурування

Загальновідомо, що оптимізаційні комбінаторні задачі є одними з найбільш важких з обчислювальної точки зору. Універсальний метод − повний перебір варіантів − застосовний практично для задач малої вимірності. Тому при розв'язанні практичної задачі часто виникає необхідність розробляти нові та удосконалювати існуючі методи, як точні, так і наближені, які враховували б специфіку цільової функції і додаткових обмежень і були б застосовні до задач більшої вимірності, ніж метод повного перебору.

Очевидно, що швидкого поширення набувають алгоритми, які краще і простіше враховують природу класів задач, що розв'язуються. В процесі розробки і реалізації алгоритму природним чином розкриваються властивості, що відображають багатокритеріальність оптимізаційних задач, а також комбінаторні характеристики, які використовувалися в модифікації з новими підходами.

За останні роки в Інституті кібернетики імені В. М. Глушкова НАН України розроблено новий, так званий **метод направленого структурування**, для розв'язування подібних задач.

Основна ідея запропонованого методу перекликається з відомим методом послідовного аналізу варіантів, суть якого описано в першому розділі. Але новий метод призначений безпосередньо для комбінаторних задач. Як відомо, у більшості задач на комбінаторних конфігураціях постає необхідність перераховувати велику кількість варіантів, порівнянну з факторіальною величиною. Тому, метод об'єднує засоби комбінаторного аналі-

зу та теорії графів і передбачає послідовне виконання трьох стадій:

1. Вибір способу генерування у певній послідовності всіх елементів заданої комбінаторної конфігурації, який найбільше пристосований до заданої функції цілі;

2. Представлення множини комбінаторної конфігурації у вигляді орієнтованого графа, де дуга відповідає спаданню значень цільової функції;

3. Побудову поліноміального алгоритму розв'язку задачі на частково упорядкованих вершинах графа.

Завдяки частковій упорядкованості часто вдається трудомісткість алгоритмів звести до логарифма від всієї маси варіантів.

Цей метод було застосовано для розв'язання задач з лінійною та дробово-лінійною цільовими функціями на перестановках, розбиттях, комбінаціях та розміщеннях [56–63]. Створені відповідні інформаційні технології підтвердили ефективність цього методу. Продовжуються дослідження з метою створення відповідних алгоритмів для розв'язання комбінаторних задач з іншими цільовими функціями та на інших комбінаторних конфігураціях.

Висновки до розділу

В розділі зроблена порівняльна характеристика існуючих методів генерування комбінаторних конфігурацій – лексикографічного, антилексикографічного та розглянуто метод генерування всіх перестановок за мінімальне число транспозицій. Побудовано та описано нові методи генерування різних комбінаторних конфігурацій – рекурсивний метод та метод переміщення максимального елемента. В залежності від властивостей задачі визначається той чи інший метод генерування.

Нові методи генерування дають можливість представити множину комбінаторних конфігурацій у вигляді графа комбінаторних конфігурацій, який відображає елементи конфігурації і по якому можна рухатися в залежності від значення функції. Граф комбінаторної конфігурації в подальшому буде застосовано для побудови методів розв'язання екстремальних комбінаторних задач.

На основі графа генерування запропоновано та описано схему нового методу направленого структурування.

Отже, враховуючи вищезазначене, для розв'язування складних екстремальних задач було розроблено новий, так званий метод направленого структурування. Його основна ідея перекликається з відомим методом послідовного аналізу варіантів, але спеціально пристосованого для розв'язування комбінаторних задач. Як відомо, у більшості задач на комбінаторних конфігураціях постає необхідність перераховувати велику кількість варіантів, порівняну з факторіальною величиною. Метод об'єднує засоби комбінаторного аналізу та теорії графів і дає можливість обійти великий перебір елементів конфігурації, на якій розглядається задача.