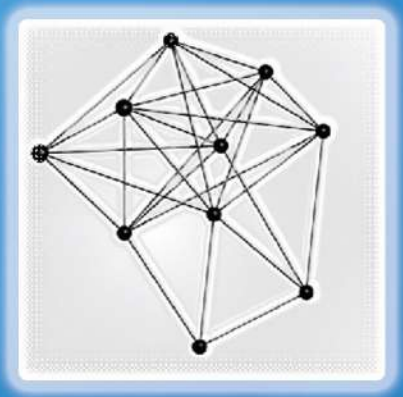
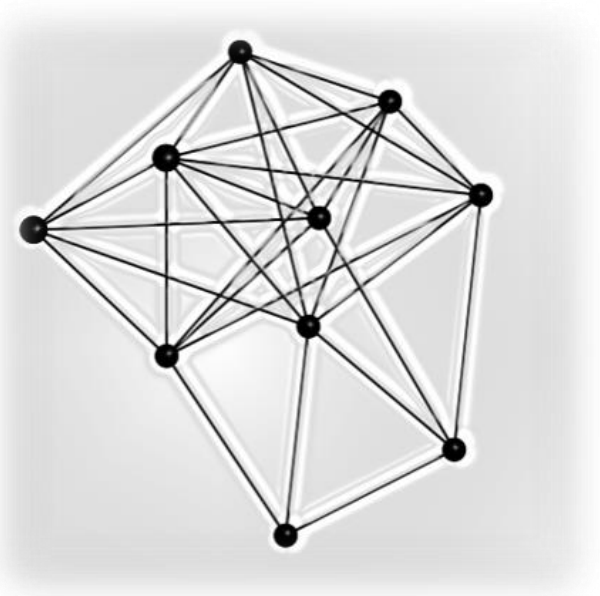
МЕТОДИ НЕГЛАДКОЇ ОПТИМІЗАЦІЇ В ПРИКЛАДНИХ ЗАДАЧАХ



Київ - 2023

МЕТОДИ НЕГЛАДКОЇ ОПТИМІЗАЦІЇ В ПРИКЛАДНИХ ЗАДАЧАХ



Київ – 2023

УДК 519.8

Автори:
П. І. Стецюк, М. Ю. Григорак, В. О. Стовба, О. А. Березовський,
Т. В. Бєлих, Є. А. Блохін, О. І. Воловик, А. А. Гонта, В. О. Жидков,
О. О. Жмуд, М. Г. Журбенко, О. П. Лиховид, В. В. Савицький,
В. П. Сизоненко, А. А. Супрун, О. М. Хом'як

Відповідальні редактори:
доктор фізико-математичних наук Петро Іванович Стецюк
(розділи 1 і 3)
доктор економічних наук Марія Юріївна Григорак
(розділ 2)

Відповідальний за макет:
доктор філософії Віктор Олександрович Стовба

Методи негладкої оптимізації в прикладних задачах / Стецюк П.І., М.Ю. Григорак, В.О. Стовба та ін. Київ: ЛАЗУРИТ ПОЛІГРАФ, 2023. 382 с.

Описано методи негладкої оптимізації та їх застосування для розв'язання прикладних задач транспорту, логістики, екології тощо. Наведено програмні реалізації описаних методів мовами Ratfor та AMPL. Розглядаються методи розв'язання задач знаходження оптимальних маршрутів та модернізації мереж, методи підтримки прийняття рішень з урахуванням ESG-чинників, а також використання r-алгоритму в прикладних задачах оптимізації.

Книга розрахована на фахівців у галузі математичного програмування, студентів та аспірантів відповідних спеціальностей.

Рецензенти:
П. С. Кнопов, чл.-кор. НАН України
В. М. Горбачук, д-р фіз.-мат. наук
В. А. Заславський, д-р техн. наук

Рекомендовано до друку Вченою радою
Інституту кібернетики імені В.М. Глушкова НАН України
(Протокол №8 від 27 червня 2023 року)

Публікація книги здійснена за фінансової підтримки
НАН України (проєкти № 0122U000833 та № 0121U100459)
та Volkswagen Foundation (грант № 97 775)

ISBN 978-617-8268-00-2

ЗМІСТ

ВСТУП 9

РОЗДІЛ 1. ЗАДАЧІ ЗНАХОДЖЕННЯ ОПТИМАЛЬНИХ МАРШРУТІВ ТА МОДЕРНІЗАЦІЇ МЕРЕЖ

1.1 Оптимізаційні задачі модернізації пропускних спроможностей дуг відмовостійких мереж 11

П. І. Стецюк, О. П. Лиховид, В. О. Жидков, А. А. Супрун

1.1.1 Вступ 11
1.1.2 Основні поняття для відмовостійкої мережі 13
1.1.3 Базові ЛП-задачі A та P 17
1.1.4 Булеві задачі A та P 21
1.1.5 Опуклі задачі A та P. Декомпозиційні алгоритми 25
1.1.6 Програмне забезпечення: програми SolverA та SolverP ... 29
1.1.7 Висновки 32
Список літератури 33

1.2 Багатопродуктова транспортна мережа з урахуванням модернізації мережі 35

М. Г. Журбенко

1.2.1 Вступ 35
1.2.2 Математична модель 35
1.2.3 Транспортна мережа 36
1.2.4 Кореспонденції перевезень 36
1.2.5 Пропускна спроможність ділянок транспортної мережі .. 39
1.2.6 Метод розв’язання 40
1.2.7 Програмне забезпечення 44
1.2.8 Висновки 45
Список літератури 46

1.3 Задачі про найкоротші k-вершинні цикли та шляхи 47

П. І. Стецюк, О. О. Жмуд

1.3.1 Вступ 47
1.3.2 Два формулювання задачі про найкоротший k-вершинний цикл 48
1.3.3 Задача комівояжера і обчислювальні експерименти 53
1.3.4 Задача про найкоротший k-вершинний шлях 57
1.3.5 Оптимальні k-маршрути для Малопольського винного шляху 60
1.3.6 Висновки 64
Список літератури 65

1.4 Двоетапна транспортна задача з обмеженням на кількість проміжних пунктів 68

П. І. Стецюк, О. М. Хом'як, О. О. Жмуд, А. А. Супрун

1.4.1 Вступ 68
1.4.2 Формулювання задачі 70
1.4.3 Про властивості задачі 73
1.4.4 AMPL-реалізація задачі 74
1.4.5 Тестовий приклад 75
1.4.6 Висновки 79
Список літератури 80

РОЗДІЛ 2. МАТЕМАТИЧНІ МЕТОДИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ З УРАХУВАННЯМ ESG-ЧИННИКІВ

2.1 Методологія багатофакторної оцінки інвестиційних проєктів сталого розвитку національної економіки 82

А. А. Гонта, М. Ю. Григорак, О. І. Воловик

2.1.1 Вступ 82
2.1.2 Взаємозв'язок та взаємовпливи кредитних

та ESG-ризиків 84
2.1.3 Постановка задачі управління ESG-ризиків в умовах цифровизації бізнесу 87
2.1.4 Розробка процедури скринінгу та скорингу зелених інвестиційних проєктів 90
2.1.5 Аналіз математичного апарату якісного і кількісного оцінювання зелених інвестиційних проєктів 95
2.1.6 Висновки 102
Список літератури 103

2.2 Статистичні та оптимізаційні методи в кредитному скорингу 107

В. О. Стовба

2.2.1 Вступ 107
2.2.2 Кредитний скоринг: основні поняття та процеси 108
2.2.3 Статистичні методи для побудови кредитних скорингових карток 109
2.2.4 Нестатистичні методи для побудови кредитних скорингових карток 118
2.2.5 Порівняння методів 132
2.2.6 Висновки 133
Список літератури 134

2.3 Методи оптимізації рішень в зелених ланцюгах постачання 138

М. Ю. Григорак

2.3.1 Вступ 138
2.3.2 Постановка задачі оптимізації викидів парникових газів в зелених ланцюгах постачання 142
2.3.3 Адаптовані моделі мережевих потоків з мінімальною вартістю для мінімізації викидів парникових газів 149

2.3.4 Метод DEA для оцінювання ефективності управлінських рішень в зелених ланцюгах постачання з орієнтацією на еталонні значення (бенчмарки) 155
2.3.5 Висновки 164
Список літератури 165

2.4 Математичні моделі, методи та програмне забезпечення для планування міжгалузевих структурно-технологічних змін 168

П. І. Стецюк, О. А. Березовський, О. П. Лиховид

2.4.1 Вступ 169
2.4.2 Базові відомості 170
2.4.3 Постановка задачі 173
2.4.4 Оптимізаційні задачі та їхній аналіз 175
2.4.5 Методи розв'язування задач максимізації D і k 179
2.4.6 Розширені оптимізаційні задачі 183
2.4.7 Тестові експерименти (Mulstr1) 190
2.4.8 Міжгалузева модель структурно-технологічних змін з індексами цін 193
2.4.9 Модифікована міжгалузева модель структурно-технологічних змін 197
2.4.10 Висновки 203
Список літератури 204

РОЗДІЛ 3. r-АЛГОРИТМ В ПРИКЛАДНИХ ЗАДАЧАХ ОПТИМІЗАЦІЇ

3.1 Оптимізаційні задачі для максимального k-плекса 206

П. І. Стецюк, О. М. Хом'як, А. А. Супрун

3.1.1 Вступ 206
3.1.2 Загальні відомості про k-плекс 207
3.1.3 Квадратичні обмеження для k-плекса 209

3.1.4 Квадратична булева задача для $\rho_k(G)$ 212
3.1.5 Максимальний 1-плекс та максимальна кліка 216
3.1.6 Уточнення лагранжевих оцінок для максимального 2-плекса 221
3.1.7 Застосування GLPK для знаходження всіх розв'язків ... 224
3.1.8 Висновки 227
Список літератури 228

3.2 Пошук дефектів у регулярних 3D-структурах 230

П. І. Стецюк, В. В. Савицький, В. О. Жидков

3.2.1 Вступ 230
3.2.2 Регулярні 3D-структури, їхні параметри та дефекти 232
3.2.3 Задачі для пошуку найкращих параметрів 236
3.2.4 Методи найменших квадратів та найменших модулів ... 242
3.2.5 Про нерівності для суми квадратів двох наборів чисел . 249
3.2.6 Висновки 254
Список літератури 256

3.3 Побудова S-подібної параметричної кривої та її застосування для проєктування зовнішнього контура сопла 258

П. І. Стецюк, О. М. Хом'як, В. О. Жидков

3.3.1 Вступ 259
3.3.2 Геометрична модель задачі 259
3.3.3 Система нелінійних інтегральних рівнянь. Її властивості 261
3.3.4 Оптимізаційна задача та алгоритм її розв'язання 267
3.3.5 Три обчислювальні експерименти 273
3.3.6 Геометричне моделювання зовнішнього контуру сопла Франкля 278
3.3.7 Опис програмного забезпечення 282

3.3.8 Висновки 290
Список літератури 291
3.4 Застосування моделі UNDBE у поєднанні з методом RALG при вирішенні задач радіоекології водних об'єктів 293
Т. В. Бєлих, В. П. Сизоненко
3.4.1 Вступ 293
3.4.2 Математична модель та комп'ютерна реалізація 295
3.4.3 Моделювання переносу тритію в руслі р. Луари 301
3.4.4 Моделювання переносу стронцію-90 в Київському водосховищі 305
3.4.5 Висновки 309
Список літератури 309
ДОДАТОК А. Ratfor програма SolverA 311
ДОДАТОК Б. k-Вершинний цикл: AMPL-реалізація задач 333
ДОДАТОК В. Ratfor програма Mulstr1 343
ДОДАТОК Г. Ratfor програма Mulstr2 364

ВСТУП

У книзі представлено ряд результатів наукових досліджень відділу методів негладкої оптимізації Інституту кібернетики імені В.М. Глушкова НАН України, які пов'язані з застосуванням методів негладкої оптимізації в прикладних задачах. Ці результати отримано в період 2018–2022 рр. в результаті виконання науково-дослідних робіт за фундаментальними та прикладними темами Національної академії наук України (проєкт № 0116U004558 «Розробити програмні засоби для субградієнтних алгоритмів з перетворенням простору та прикладних задач оптимізації», 2016–2018 рр.; проєкт № 0117U002494 «Розробити нові математичні моделі та методи для створення комп'ютерних технологій розв'язання складних задач прикладної математики», 2017–2021 рр.; проєкт № 0117U000327 «Розробити субградієнтні алгоритми розв'язання багатоекстремальних квадратичних оптимізаційних задач», 2017–2021 рр.; проєкт № 0119U001641 «Розроблення оптимізаційних процедур для задач розташування накопичувачів електроенергії в ОЕС України в сучасних умовах технологічних змін. 1 етап. Розроблення математичних моделей, методів та програмного забезпечення для спеціальних класів двоетапних транспортних задач», 2019 р.; проєкт № 0121U100459 «Розробити методи негладкої оптимізації для побудови кривих у натуральній параметризації», 2021–2023 рр.) та грантом CRDF Global та МОН України (G-202102-68020 «Методи оптимізації зі зменшенням ризиків для розміщення об'єктів у виробництві відновлюваної енергії», 2021–2022 рр.).

Розділ 1 присвячено задачам знаходження оптимальних маршрутів та модернізації мереж. У підрозділі 1.1 розглядаються оптимізаційні задачі модернізації пропускних спроможностей дуг відмовостійких мереж; у підрозділі 1.2 розглядається багатопродуктова транспортна задача з урахуванням модернізації мереж; у підрозділі 1.3 розглядаються задачі про найкоротші k-вершинні цикли та шляхи; у підрозділі 1.4 розглядається двоетапна

транспортна задача з обмеженням на кількість проміжних пунктів. У додатках А та Б наведено Ratfor та AMPL програми, що реалізують відповідні моделі та методи, а також результати їхньої роботи.

У розділі 2 розглядаються математичні методи підтримки прийняття рішень з урахуванням ESG-чинників (Environmental, Social, and Corporate Governance). У підрозділі 2.1 досліджується методологія багатофакторної оцінки інвестиційних проєктів сталого розвитку національної економіки; у підрозділі 2.2 розглядаються статистичні та оптимізаційні методи в кредитному скорингу; у підрозділі 2.3 розглядаються методи оптимізації рішень в зелених ланцюгах постачання; у підрозділі 2.4 досліджуються математичні моделі, методи та програмне забезпечення для планування міжгалузевих структурно-технологічних змін.

У розділі 3 представлено результати застосування модифікацій r-алгоритму для різноманітних прикладних задач оптимізації. У підрозділі 3.1 розглядаються оптимізаційні задачі для знаходження максимального k-плекса у неорієнтованому графі; у підрозділі 3.2 досліджується задача пошуку дефектів у регулярних 3D-структурах; у підрозділі 3.3 розглядається задача побудови S-подібної параметричної кривої та її застосування для проєктування зовнішнього контура сопла; у підрозділі 3.4 описано застосування методу RALG для розв’язання задач радіоекології водних об’єктів.

Представлені в книзі методи можуть використовуватись при розв’язанні широкого кола спеціальних задач в області дослідження операцій, які виникають при прийнятті оптимальних рішень в управлінні, плануванні, проєктуванні, моделюванні тощо. Розроблені програмні засоби можуть використовуватись як для розв’язання конкретних прикладних задач, так і в навчальному процесі для підготовки студентів відповідних спеціальностей.

Розділ 1. ЗАДАЧІ ЗНАХОДЖЕННЯ ОПТИМАЛЬНИХ МАРШРУТІВ ТА МОДЕРНІЗАЦІЇ МЕРЕЖ

1.1 ОПТИМІЗАЦІЙНІ ЗАДАЧІ МОДЕРНІЗАЦІЇ ПРОПУСКНИХ СПРОМОЖНОСТЕЙ ДУГ ВІДМОВОСТІЙКИХ МЕРЕЖ

П. І. Стецюк, О. П. Лиховид, В. О. Жидков, А. А. Супрун

Анотація. Розглядаються оптимізаційні задачі лінійного програмування, змішаного булевого лінійного програмування та нелінійного програмування для знаходження пропускних спроможностей дуг відмовостійких мереж. Наведено результати обчислювальних експериментів розв`язання булевих задач за допомогою відомої програми Gurobi. Розроблено декомпозиційні методи на основі *r*-алгоритму Шора та показано їх конкурентоспроможність з програмою IPOPT при розв’язанні задач нелінійного програмування.

Abstract. Optimization problems of linear programming, mixed Boolean linear programming, and nonlinear programming are considered for finding the bandwidth capacities of arcs of fault-tolerant networks. The results of computational experiments for solving Boolean problems using the well-known Gurobi program are presented. Decomposition methods based on Shor's r-algorithm are developed and their competitiveness with the IPOPT program in solving nonlinear programming problems is shown.

1.1.1 Вступ

Задачі знаходження як структур, так і параметрів стійких і безвідмовних мереж (телекомунікаційних, комп'ютерних, транспортних, енергетичних тощо) є нині актуальними [1–4].

Проблемам побудови оптимізаційних задач для відмовостійких мереж присвячена робота [5], де розглядаються такі важливі з точки зору практичних потреб задачі на мережах: проектування мережі мінімальної вартості за умови виходу з ладу окремих ланок, знаходження пропускних спроможностей дуг надійної орієнтованої мережі, проектування оптимальної логічної структури надійної мережі, задача модернізації надійної мережі, задача оптимізації мереж з урахуванням неповної інформації, перспективне планування перевезень, знаходження оптимальної номенклатури рухомого складу. Розглянуто задачі лінійного програмування (ЛП-задачі) для знаходження оптимальних пропускних спроможностей дуг відмовостійкої орієнтованої мережі. У статтях [6, 7] ці ЛП-задачі узагальнювались на випадок нелінійних опуклих задач, а у [8, 9] на випадок булевих лінійних задач.

У даному підрозділі, матеріал якого базується на роботі [10], розглянуто математичні моделі двох класів задач знаходження пропускних спроможностей дуг відмовостійкої орієнтованої мережі. Відмовостійкою вважається мережа, для якої можна задовольнити всі вимоги на передачу потоків як при відсутності відмов у мережі, так і тоді, коли буде мати місце будь-яка одна з усіх можливих одиничних відмов мережі. У першому класі задач (задача А) для передачі потоків можуть використовуватись всі можливі шляхи в мережі. У другому класі задач (задача Р) для передачі потоків задіяні тільки шляхи із наперед заданої множини шляхів. Математичні моделі представлені задачами лінійного, змішаного булевого лінійного та нелінійного програмування з блочною структурою матриці обмежень.

Матеріал підрозділу представлено в такому порядку. Описані поняття одиничної відмови та сценарію відмов мережі, наведено зміст оптимізаційних задач А та Р для модернізації пропускних спроможностей дуг відмовостійкої мережі, описано тестову мережу (6 вершин та 19 дуг) для перевірки алгоритмів розв’язання задач модернізації відмовостійких мереж. Описані базові моделі задач лінійного програмування для знаходження пропускних спроможностей дуг відмовостійкої фізичної структури мережі

(задача А) та відмовостійкої логічної структури мережі (задача Р), розглянуто їх властивості. Описано задачі А та Р у формі моделей змішаного булевого лінійного програмування. Наведено оптимальні розв'язки задачі А для різних сценаріїв відмов у тестовій мережі. Розв'язки знайдені за допомогою програми Gurobi [11] з NEOS-сервера [12], де математична модель задачі А описана мовою моделювання AMPL [13]. Описано нелінійні моделі опуклого програмування для задач А та Р, які призначено для знаходження оптимальних (за вибраним критерієм) пропускних спроможностей дуг відмостійких мереж, та описано декомпозиційний алгоритм їх розвязання. Наведено опис програмного забезпечення для декомпозиційного алгоритму на основі ефективних реалізацій r-алгоритмів Шора [14–16]. Проведено порівняння програми IPOPT [17] та декомпозиційного алгоритму для розв'язання тестових задач.

1.1.2 Основні поняття для відмовостійкої мережі

Нехай $N=(V,A)$ – орієнтована мережа з множиною вершин V та множиною дуг A. Пропускну спроможність дуги $a=(i,j)\in A$, спрямованої з вершини $i\in V$ у вершину $j\in V$, позначимо y_a.

Означення. *Одиничною відмовою мережі назвемо такий її стан, коли пропускні спроможності дуг змінюються за правилом:*

$$y_a^{'}=\mu_a y_a,\ \forall a\in A,$$

якщо хоча б один з коефіцієнтів $\mu_a\in[0,1)$.

За допомогою цього поняття можна описати різноманітні аварійні ситуації, що мають місце у функціонуючих мережах будь-якого типу: автомобільних, залізничних, комунікаційних, енергетичних тощо. Множину одиничних відмов мережі будемо називати сценарієм відмов мережі та будемо її позначати буквою F (**F**ault) в комбінації з відповідним цьому сценарію набором цифр. Зауважимо, що для зручності одиничною відмовою мережі будемо вважати її повноцінне

функціонування, що буде мати місце, якщо для всіх дуг коефіцієнти $\mu_a = 1$.

Для мережі $Net(4,4)$ (рисунок 1.1.1) в таблиці 1.1.1 наведено сценарій відмов 0,5F, коли одна, але будь-яка дуга з чотирьох зменшує свою пропускну спроможність у два рази ($\mu_a = 0.5$). Якщо для сценарію 0,5F всі значення 0,5 замінити нульовими, то отримаємо новий сценарій відмов 1F, рівносильний повній відмові тільки однієї будь-якої дуги в мережі. На рисунку 1.1.2 наведено одиничні відмови (окремі пошкодження) у мережі $Net(4,4)$ для сценарю 1F.

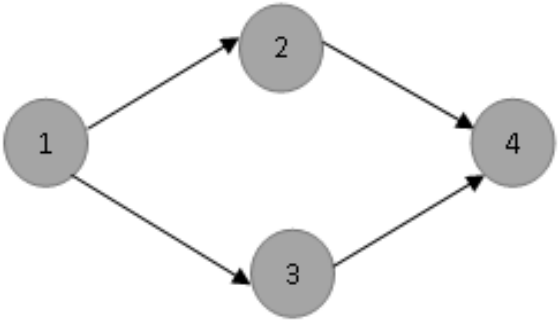


Рис. 1.1.1. Мережа $Net(4,4)$, 4 вершини і 4 дуги

Табл. 1.1.1. Сценарії 0F; 0,5F; 1F для мережі $Net(4,4)$

№	Дуги	0F	0,5F				1F			
	(i,j)	μ_0	μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7	μ_8
1	$(1,2)$	1,0	0,5	1,0	1,0	1,0	0,0	1,0	1,0	1,0
2	$(1,3)$	1,0	1,0	0,5	1,0	1,0	1,0	0,0	1,0	1,0
3	$(2,4)$	1,0	1,0	1,0	0,5	1,0	1,0	1,0	0,0	1,0
4	$(3,4)$	1,0	1,0	1,0	1,0	0,5	1,0	1,0	1,0	0,0

Зміст задач знаходження пропускних спроможностей дуг та вершин для відмовостійкої мережі полягає в наступному.

Задані:

1. Мережа $N=(V,A)$**;** y_a^0 – значення існуючих пропускних спроможностей дуг, $a\in A$; y_a^{low}, y_a^{up} – нижня та верхня межі на пропускні спроможності дуг, $a\in A$.

2. Мережевий трафік K – обсяги потоку в мережі; d_k, $k\in K$ – обсяг потоку від вершини $s(k)\in V$ до вершини $r(k)\in V$, які будемо називати джерелом (sender) та стоком (receiver), відповідно.

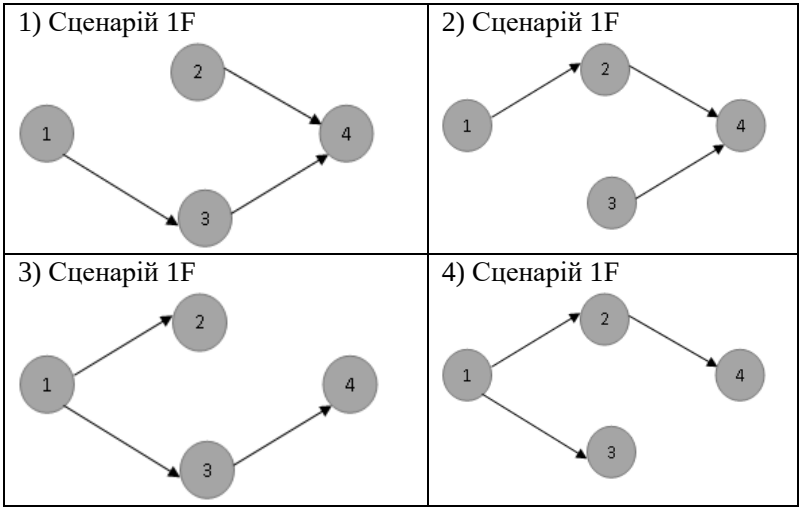


Рис. 1.1.2. Одиничні відмови мережі $Net(4,4)$ для сценарію 1F

3. Сценарії відмов T для мережі $N=(V,A)$:

μ_{at}, $\mu_{at}\in[0,1]$, $\forall a\in A$, $\forall t\in T$. Якщо $\mu=0$, то це рівнозначно відмові дуги a. Якщо $\mu=1$, то це означає що дуга a не вімовляє.

Потрібно знайти: оптимальні за критерієм (розглянемо його пізніше) модернізації пропускні спроможності дуг y_a^*, $a \in A$ (що додаються до вже існуючих y_a^0), при яких забезпечується заданий обсяг трафіку K в мережі $N = (V, A)$, якщо станеться одна, але довільна, одинична відмова зі сценарію відмов T.

Для мережі $N = (V, A)$ будемо розглядати два типи задач знаходження пропускних спроможностей дуг.

Задача А: для передачі потоків залучаються всі можливі шляхи в мережі.

Задача Р: для передачі потоків залучаються лише шляхи із заданої множини шляхів P. Тут $P = \bigcup_{k \in K} P_k$, де P_k – множина шляхів для потоку k.

Іншими словами, пересилання потоків в задачі А визначається фізичною (визначена вершинами та дугами орієнтовної мережі) структурою мережі, а в задачі Р визначається логічною (визначена набором можливих шляхів) структурою мережі.

Далі опишемо базові задачі лінійного програмування (ЛП-задачі А та Р), які будуть використані при побудові для задач А та Р відповідних моделей булевого лінійного програмування та нелінійного програмування. Розв'язки задач модернізації пропускних спроможностей дуг буде проілюстровано на прикладі орієнтованої мережі $Net(6,16)$ з шістьма вершинами та 16 дугами (рисунок 1.1.3), яка розглядалася при описі математичного забезпечення для задач проектування надійних мереж [5].

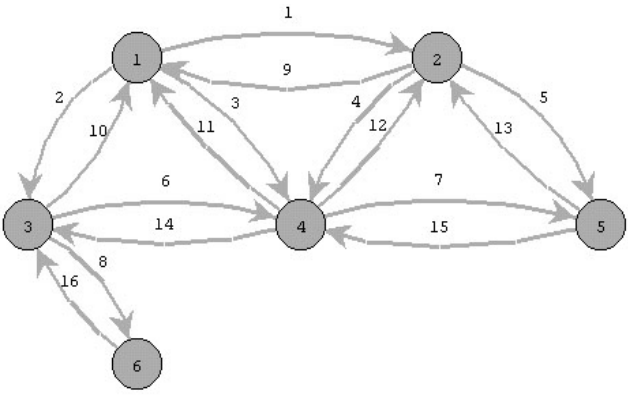


Рис. 1.1.3. Структура мережі $Net(6,16)$

Вартості створення одиниці пропускної спроможності для дуг (1,2) та (2,1) будуть рівними 1.5, а для всіх інших дуг – рівними 1. Будемо вважати, що між усіма парами вершин потрібно переслати по мережі один і той же обсяг потоку, що дорівнює десяти одиницям. Які із сценаріїв відмов у мережі $Net(6,16)$ із книги [5, ст. 114] будуть використані, будемо уточнювати для кожного окремого розрахунку тієї чи іншої задачі.

1.1.3 Базові ЛП-задачі А та Р

Нехай c_v, c_a – вартість створення одиниці пропускних спроможностей для вершини v та дуги a, A_i^+ та A_i^- – множини дуг, що входять і виходять з вершини v, $v \in V$. Тоді задача А має такий вигляд: знайти

$$c_a^* = \min_{x,y} \sum_{a \in A} c_a y_a \qquad (1.1.1)$$

за обмежень

$$\sum_{k \in K} x_{akt} \le \mu_{at} \left(y_a^0 + y_a \right), \quad \forall a \in A, \ \forall t \in T, \qquad (1.1.2)$$

$$\sum_{a\in A_i^+} x_{akt} - \sum_{a\in A_i^-} x_{akt} = \begin{cases} d_k, & \text{якщо } i = s(k); \quad \forall i \in V, \\ -d_k, & \text{якщо } i = r(k); \quad \forall k \in K, \\ 0 \text{ у протилежному випадку}; & \quad \forall t \in T, \end{cases} \quad (1.1.3)$$

$$x_{akt} \geq 0, \quad \forall a \in A, \ \forall k \in K, \ \forall t \in T, \quad (1.1.4)$$

$$y_a^{low} \leq y_a \leq y_a^{up}, \quad \forall a \in A. \quad (1.1.5)$$

Тут змінні x_{akt} позначають невідомий потік продукту k по дузі a при пошкодженні t, $s(k) = sender(k)$, $r(k) = receiver(k)$. Змінні y_a позначають невідомі значення пропускних спроможностей дуг $a \in A$, що додаються до вже існуючих пропускних спроможностей y_a^0.

Формулювання задачі Р має такий вигляд: знайти

$$c_P^* = \min_{z,y} \sum_{a\in A} c_a y_a \quad (1.1.6)$$

за обмежень

$$\sum_{k\in K} \sum_{p\in P_k} \delta_{kpa} z_{kpt} \leq \mu_{at} \left(y_a^0 + y_a \right), \quad \forall a \in A, \ \forall t \in T, \quad (1.1.7)$$

$$\sum_{p\in P_k} z_{kpt} = d_k \quad \forall k \in K, \ \forall t \in T, \quad (1.1.8)$$

$$z_{kpt} \geq 0, \quad \forall k \in K, \ \forall p \in P, \ \forall t \in T, \quad (1.1.9)$$

$$y_a^{low} \leq y_a \leq y_a^{up}, \quad \forall a \in A, \quad (1.1.10)$$

де

$$\delta_{kpa} = \begin{cases} 1, & \text{коли шлях } p \in P_k \text{ включає дугу } a; \\ 0 & \text{у протилежному випадку.} \end{cases}$$

Тут змінні z_{kpt} позначають потік продукту $k \in K$ по шляху $p \in P_k$ при пошкодженні $t \in T$. Змінні y_a такі ж як і для задачі (1.1.1) – (1.1.5). Зміст ЛП-задач (1.1.1) – (1.1.5) та (1.1.6) – (1.1.10) досить прозорий. Цільові функції (1.1.1) та (1.1.6), які мінімізуються, задають сумарні витрати на вартість тих пропускних спроможностей (що доповнюють вже існуючі) дуг, які потрібно знайти з метою забезпечення безвідмовної роботи мережі $N(V, A)$. Зміст обмежень у

ЛП-задачах А та Р такий: обмеження (1.1.2) й (1.1.7) означають, що потоки по дугах не повинні перевищувати пропускних спроможностей дуг при довільній одній одиничній відмові зі сценарію відмов; обмеження (1.1.3) й (1.1.8) гарантують виконання умов по мережевому трафіку; обмеження (1.1.4) й (1.1.9) відповідають за невід'ємність потоків, а нерівності (1.1.5) й (1.1.10) накладають двосторонні обмеження на вибір пропускних спроможностей дуг, що додаються до уже існуючих.

Для ЛП-задачі (1.1.1) – (1.1.5) кількість змінних N_A та кількість обмежень M_A (без урахування найпростіших обмежень (1.1.4) – обмежень на невід'ємність змінних x_{akt}) визначаються за формулами

$$N_A = |A|*|K|*|T|+|A|, \quad M_A = |A|*|T|+|V|*|K|*|T|+2*|A|, \quad (1.1.11)$$

де для N_A перший доданок задає кількість змінних x_{akt}, другий – кількість змінних y_a, а для M_A перший доданок задає кількість обмежень (1.1.2), другий доданок – кількість обмежень (1.1.3), третій доданок – кількість обмежень (1.1.5).

Для ЛП-задачі (1.1.6) – (1.1.10) кількість змінних N_P та кількість обмежень M_P (без урахування найпростіших обмежень (1.1.9) – обмежень на невід'ємність змінних z_{kpt}) визначається за формулами

$$N_P = |K|*|P|*|T|+|A|, \quad M_P = |A|*|T|+|K|*|T|+2*|A|, \quad (1.1.12)$$

де для N_P перший доданок задає кількість змінних z_{kpt}, другий – кількість змінних y_a, а для M_P перший доданок задає кількість обмежень (1.1.7), другий доданок – кількість обмежень (1.1.8), третій доданок – кількість обмежень (1.1.10).

Якщо мережа $N(V,A)$ має відносно невеликі розміри (20 вершин, 100 дуг), то ЛП задачі А та Р мають дуже велику вимірність. Так, наприклад, якщо $|A| \approx 100$, $|K| \approx 400$, $|V| \approx 20$, $|T| \approx 1000$, то у ЛП-задачі А кількість змінних N_A та кількість обмежень M_A мають порядок 10^7. Якщо при цьому ще й $|P| \approx 10000$, то у ЛП-задачі Р

кількість змінних N_P має порядок 10^8 та кількість обмежень M_P мають порядок 10^5.

У цьому випадку ЛП-задачі (1.1.1) – (1.1.5) та (1.1.6) – (1.1.10) неможливо розв’язати навіть за допомогою найкращих сучасних програм для задач лінійного програмування, наприклад CPLEX та Gurobi, тому що це вимагає дуже значних обчислювальних ресурсів. Однак в обох ЛП-задачах матриця обмежень має вкладену блочну структуру (рисунок 1.1.4, нульові блоки помічено білим кольором), що дає можливість розробляти декомпозиційні алгоритми розв'язування ЛП-задач А та Р і для більших, ніж наведені вище, розмірів мережі, наприклад, мережа може містити 100 вершин та 1000 дуг.

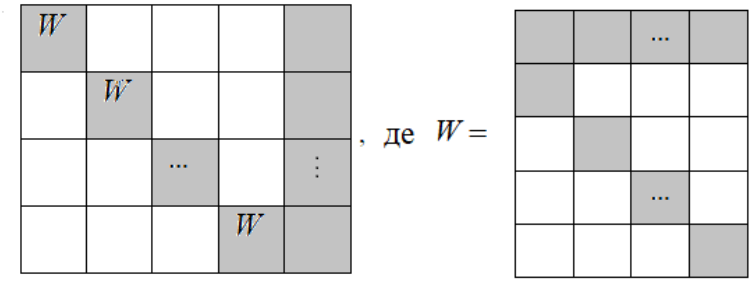


Рис. 1.1.4. Вкладена блочна структура матриці обмежень ЛП-задач А та Р

Такі декомпозиційні алгоритми будуть описані далі як для лінійної цільової функції, так і для опуклих гладкої та негладкої цільових функцій.

Нижче для задач А та Р описуються моделі змішаного булевого лінійного програмування, які враховують витрати на створення нових дуг та визначають ті дуги, пропускні спроможності яких потрібно модернізувати у відмовостійкій мережі.

1.1.4 Булеві задачі А та Р

Математичні моделі відповідних задач змішаного булевого лінійного програмування (булевих задач А та Р) легко отримати, якщо у задачах (1.1.1) – (1.1.5) та (1.1.6) – (1.1.10) цільові функції (1.1.1) і (1.1.6) замінити на цільові функції

$$F_A^* = \min_{x,y,u} \sum_{a\in A} C_a u_a + \sum_{a\in A} c_a y_a \,, \tag{1.1.13}$$

$$F_P^* = \min_{z,y,u} \sum_{a\in A} C_a u_a + \sum_{a\in A} c_a y_a \,, \tag{1.1.14}$$

а обмеження (1.1.5) та (1.1.10) замінити на обмеження

$$y_a^{low} u_a \le y_a \le y_a^{up} u_a, \quad \forall a \in A, \tag{1.1.15}$$

де C_a – витрати на створення доданих пропускних спроможностей дуг, а булеві змінні $u_a = 0 \vee 1$ дорівнюють одиниці, якщо дуга $a \in A$ додається до мережі, і нулю в протилежному випадку. Тоді формулювання булевої задачі А має такий вигляд: знайти

$$F_A^* = \min_{x,y,u} \sum_{a\in A} C_a u_a + \sum_{a\in A} c_a y_a \tag{1.1.16}$$

за обмежень

$$\sum_{k\in K} x_{akt} \le \mu_{at}\left(y_a^0 + y_a\right), \quad \forall a \in A, \ \forall t \in T \,, \tag{1.1.17}$$

$$\sum_{a\in A_i^+} x_{akt} - \sum_{a\in A_i^-} x_{akt} = \begin{cases} d_k, & \text{якщо } i = s(k); \quad \forall i \in V, \\ -d_k, & \text{якщо } i = r(k); \quad \forall k \in K, \\ 0 \text{ у протилежному випадку}; & \quad \forall t \in T, \end{cases} \tag{1.1.18}$$

$$x_{akt} \ge 0, \quad \forall a \in A, \ \forall k \in K, \ \forall t \in T \,, \tag{1.1.19}$$

$$y_a^{low} u_a \le y_a \le y_a^{up} u_a, \quad \forall a \in A. \tag{1.1.20}$$

Формулювання булевої задачі Р має такий вигляд: знайти

$$F_P^* = \min_{z,y,u} \sum_{a\in A} C_a u_a + \sum_{a\in A} c_a y_a \tag{1.1.21}$$

за обмежень

$$\sum_{k\in K}\sum_{p\in P_k}\delta_{kpa}z_{kpt} \le \mu_{at}\left(y_a^0 + y_a\right), \quad \forall a \in A,\ \forall t \in T, \quad (1.1.22)$$

$$\sum_{p\in P_k} z_{kpt} = d_k \quad \forall k \in K,\ \forall t \in T, \quad (1.1.23)$$

$$z_{kpt} \ge 0, \quad \forall k \in K,\ \forall p \in P,\ \forall t \in T, \quad (1.1.24)$$

$$y_a^{low} u_a \le y_a \le y_a^{up} u_a, \quad \forall a \in A. \quad (1.1.25)$$

Якщо $C_a = 0$, то розв'язки булевих задач (1.1.16) – (1.1.20) та (1.1.21) – (1.1.25) співпадають з розв`язками ЛП-задач (1.1.1) – (1.1.5) та (1.1.6) – (1.1.10). Якщо $C_a > 0$, то мінімізуватися будуть сумарні витрати на вартість та створення пропускних спроможностей дуг, що доповнюють уже існуючі у мережі. Вибір відповідних дуг мережі буде залежати від коефіцієнтів $C_a > 0$, $a \in A$.

Один із можливих способів розв'язання задач (1.1.16) – (1.1.20) і (1.1.21) – (1.1.25) полягає у представленні їх на мові моделювання AMPL (A Mathematical Programming Language) [13] та використанні сучасного програмного забезпечення для розв’язання задач цілочислового лінійного програмування, наприклад, програми Gurobi [11] з NEOS-сервера [12]. У таблицях 1.1.2 та 1.1.3 наведено результати тестування програми Gurobi для розв'язання задачі (1.1.16) – (1.1.20) на прикладі орієнтованої мережі Net(6,16) (рисунок 1.1.3). Тут сценарії відмов відповідають сценаріям (a) – (f) з книги [5, ст. 114–115]:

a) відсутність одиничних відмов;

b) відмовити може будь-яка, але одна дуга, за винятком дуг, зв'язаних з вершиною 6;

c) одна відмова, коли одночасно відмовляють чотири дуги: (1,2), (2,1), (2,4) та (4,2);

f) будь-яка з дуг, але тільки одна, може зменшити свою пропускну спроможність в два рази.

Табл. 1.1.2. Мінімальні за сумарною вартістю пропускні спроможності дуг мережі Net(6,16)

N	(i,j)	c_{ij}	$y_a^0 = y_{ij}^0 = 0$				$y_{ij}^0 = \left(y_{ij}^*\right)_a$		
			a	b	c	f	b	c	f
1	(1, 2)	1.5	10	50	0	46.66	40	0	20
2	(1, 3)	1	20	80	20	53.33	60	0	30
3	(1, 4)	1	20	40	30	16.66	20	10	10
4	(2, 4)	1	30	30	0	10	0	0	0
5	(2, 5)	1	10	50	50	26.66	40	40	20
6	(3, 4)	1	60	80	60	53.33	20	0	0
7	(4, 5)	1	40	50	80	46.66	10	40	0
8	(3, 6)	1	50	50	50	100	0	0	50
9	(2, 1)	1.5	10	50	0	46.66	40	0	20
10	(3, 1)	1	20	80	20	53.33	60	0	30
11	(4, 1)	1	20	40	30	16.66	20	10	10
12	(4, 2)	1	30	30	0	10	0	0	0
13	(5, 2)	1	10	50	50	26.66	40	40	20
14	(4, 3)	1	60	80	60	53.33	20	0	0
15	(5, 4)	1	40	50	80	46.66	10	40	0
16	(6, 3)	1	50	50	50	100	0	0	50
$\sum_{(i,j)\in A} c_{ij} y_{ij}^*$			490	910	580	753.33	420	180	280

В таблиці 1.1.2 наведено два варіанти мінімальних за сумарною вартістю пропускних спроможностей дуг мережі Net(6,16) при сценаріях відмов (a), (b), (c), (f), яким відповідають нульові значення $C_a = 0$. Вони отримані програмою Gurobi при двох різних значеннях існуючих пропускних спроможностей – нульових значеннях $y_a^0 = y_{ij}^0 = 0$ та ненульових значеннях $y_{ij}^0 = \left(y_{ij}^*\right)_a$, які отримано для

сценарію (а) при існуючих пропускних спроможностях $y_a^0 = y_{ij}^0 = 0$. При цьому час розв'язання дорівнював декільком секундам.

Зауважимо, що результати розрахунку в таблиці 1.1.2 співпадають з результатами розрахунків із книги [5, ст. 116–117], що і повинно бути, так як ЛП-задачі А із книги [5], співпадають з ЛП-задачею (1.1.1) – (1.1.5).

Табл. 1.1.3. Мінімальні за сумарними витратами на вартість та створення пропускні спроможності дуг мережі Net(6,16)

N	(i, j)	C_a	Без активних обмежень (15а)				Активна дуга $y_{11}^{up} = 20$		
			a	b	c	f	a	b	C
1	(1, 2)	10	0	50	0	33.33	0	50	0
2	(1, 3)	10	20	80	20	43.33	20	80	20
3	(1, 4)	10	30	40	30	30	30	40	30
4	(2, 4)	10	40	30	0	0	40	0	0
5	(2, 5)	10	10	50	50	33.33	0	50	50
6	(3, 4)	10	60	80	60	53.33	60	80	60
7	(4, 5)	10	40	50	80	33.33	50	50	80
8	(3, 6)	10	50	50	50	100	50	50	50
9	(2, 1)	10	0	50	0	53.33	10	80	0
10	(3, 1)	10	20	80	20	53.33	20	80	30
11	(4, 1)	10	30	40	30	0	20	10	20
12	(4, 2)	10	50	30	0	30	50	30	0
13	(5, 2)	10	1.30 E-12	50	50	23.33	1.90 E-12	50	50
14	(4, 3)	10	60	80	60	73.33	60	80	70
15	(5, 4)	10	50	50	80	53.33	50	80	80
16	(6, 3)	10	50	50	50	100	50	50	50

$\sum_{a\in A_a} C_a u_a + \sum_{a\in A_a} c_a y_a$	640	1070	700	896.67	645	1075	710

В таблиці 1.1.3 наведено оптимальні (мінімальні за сумарними витратами на вартість та створення) додані пропускні спроможності дуг мережі Net(6,16) для ненульових значень $C_a = 10$ при тих же сценаріях відмов (a), (b), (c), (f). Вони отримані за допомогою програми Gurobi для варіанту без активних обмежень (1.1.20) та для варіанту з обмеженням зверху на пропускну спроможність для всього однієї дуги $y_{11}^{up} = y_{(4,1)}^{up} = 20$. Для варіанту з обмеженням зверху на пропускну спроможність в таблиці 1.1.3 відсутній варіант (f). Це пов'язано з тим, що розв'язок задачі для варіанту (f) без активних обмежень (1.1.20) повністю співпадає з розв'язком для варіанту (f) з обмеженням для дуги 11. Із таблиці 1.1.3 видно, що активне обмеження зверху на пропускну спроможність навіть однієї дуги (зменшено на 10 одиниць порівняно з зі значенням із розв'язку задачі для варіанту (a) без активних обмежень (1.1.20)) може призвести до досить суттєвого перерозподілу оптимальних пропускних спроможностей дуг (дуги 4,5,9). Час розв'язання задач для таблиці 1.1.3 також сягав декількох секунд.

1.1.5 Опуклі задачі A та P. Декомпозиційні алгоритми

Математичні моделі задач знаходження пропускних спроможностей дуг відмовостійкої мережі можуть представлені як опуклі оптимізаційні задачі, в яких мінімізується цільова опукла функція (може бути як гладкою, так і негладкою) від невідомих $Y = \{y_a,\ \forall a \in A\}$ – пропускних спроможностей дуг мережі. Частковим випадком опуклих задач є ЛП-задачі A та P, для них опукла функція є лінійною функцією $f_1(Y) = \sum_{a\in A} c_a y_a$ та мінімізує сумарні витрати за вартістю пропукних спроможностей дуг відмовостійкої мережі.

Квадратична гладка функція $f_2(Y)=\sum_{a\in A}\left(y_a-y_a^e\right)^2$ і негладка функція $f_3(Y)=\sum_{a\in A}\left|y_a-y_a^e\right|$, де y_a^e, $a\in A$, – деякі «бажані» пропускні спроможності дуг, дозволяють знайти пропускні спроможності дуг мережі з мінімальним відхиленням від «бажаних» пропускних спроможностей дуг за критерієм найменших квадратів та критерієм найменших модулів. Нижче розглянемо задачі мінімізації опуклої функції $f(Y)=f_i(Y)$, де $i\in\{1,2,3\}$, при обмеженнях ЛП-задач, прибравши в (1.1.5) та (1.1.10) двосторонні обмеження на невідомі пропускні спроможності дуг.

Тоді формулювання опуклої задачі А має такий вигляд: знайти

$$f_A^*=\min_{x,y}\left\{f(Y)+\sum_{t\in T}\sum_{a\in A}Q_a y_{at}\right\}, \qquad (1.1.26)$$

за обмежень

$$\sum_{k\in K}x_{akt}-y_{at}\le\mu_{at}\left(y_a^0+y_a\right),\quad \forall a\in A,\ \forall t\in T, \qquad (1.1.27)$$

$$\sum_{a\in A_i^+}x_{akt}-\sum_{a\in A_i^-}x_{akt}=\begin{cases}d_k, & \text{якщо } i=s(k);\quad \forall i\in V,\\ -d_k, & \text{якщо } i=r(k);\quad \forall k\in K,\\ 0 \text{ у протилежному випадку}; & \forall t\in T,\end{cases} \qquad (1.1.28)$$

$$x_{akt}\ge 0,\quad \forall a\in A,\ \forall k\in K,\ \forall t\in T, \qquad (1.1.29)$$

$$y_{at}\ge 0,\quad \forall a\in A,\ \forall t\in T,\ \ y_a\ge 0,\quad \forall a\in A. \qquad (1.1.30)$$

Формулювання опуклої задачі Р має такий вигляд: знайти

$$f_P^*=\min_{z,y}\left\{f(Y)+\sum_{t\in T}\sum_{a\in A}Q_a y_{at}\right\}, \qquad (1.1.31)$$

за обмежень

$$\sum_{k\in K}\sum_{p\in P_k}\delta_{kpa}z_{kpt}-y_{at}\le\mu_{at}\left(y_a^0+y_a\right),\quad \forall a\in A,\ \forall t\in T, \qquad (1.1.32)$$

$$\sum_{p\in P_k}z_{kpt}=d_k\quad \forall k\in K,\ \forall t\in T, \qquad (1.1.33)$$

$$z_{kpt} \geq 0, \quad \forall k \in K, \ \forall p \in P, \ \forall t \in T, \tag{1.1.34}$$

$$y_{at} \geq 0, \quad \forall a \in A, \forall t \in T, \ \ y_a \geq 0, \quad \forall a \in A. \tag{1.1.35}$$

Для забезпечення сумісності систем обмежень введено додаткові невід'ємні змінні $y_{at} \geq 0$, $\forall a \in A$, $\forall t \in T$. Вони мають такий зміст: y_{at} відповідає тому значенню пропускної спроможності дуги $a \in A$, якого не вистачає при виникненні t-ї одиничної відмови мережі для виконання обмежень (1.1.27) та (1.1.32), що відповідають цим $a \in A$ та $t \in T$. «Штрафні» множники Q_{at} вибираються такими, щоб оптимальні значення y^*_{at} дорівнювали нулю. Якщо структура одиничних відмов така, що відмовостійке функціонування мережі $N(V, A)$ неможливе, то ненульові оптимальні значення y^*_{at} будуть характеризувати ті критичні місця в мережі $N(V, A)$, через які неможливо забезпечити її відмовостійке функціонування.

З структури опуклих задач (рисунок 1.1.4) легко бачити, що змінні $Y = \{y_a, \forall a \in A\}$ є зв'язуючими в обох задачах. Тому для їх розв'язання доцільно використовувати схему декомпозиції за змінними Y [14]. При цьому координуючі (зовнішні) задачі полягають в мінімізації негладких опуклих функцій $F_A(Y)$ та $F_P(Y)$ від зв'язуючих змінних Y і для знаходження їх мінімумів можна застосовувати r-алгоритм [14–16], який вважається одним із ефективних методів негладкої оптимізації [18–20].

Внутрішня підзадача, яку потрібно розв'язати для обчислення субградієнта функції $F_A(Y)$, буде пов'язана з пошуком значень двоїстих змінних до обмежень (1.1.37) для задачі лінійного програмування такого вигляду: знайти

$$\varphi^*_A(t) = \min_{x,y} \sum_{a \in A} Q_a y_{at} \tag{1.1.36}$$

за обмежень

$$\sum_{k \in K} x_{akt} - y_{at} \leq \mu_{at} \left(y^0_a + \bar{y}_a \right), \quad \forall a \in A, \tag{1.1.37}$$

$$\sum_{a\in A_i^+} x_{akt} - \sum_{a\in A_i^-} x_{akt} = \begin{cases} d_k, & \text{якщо } i = s(k); \\ -d_k, & \text{якщо } i = r(k); \\ 0 & \text{у протилежному випадку;} \end{cases} \quad \forall i \in V,\ \forall k \in K, \quad (1.1.38)$$

$$x_{akt} \geq 0, \quad \forall a \in A,\ \forall k \in K, \quad (1.1.39)$$

$$y_{at} \geq 0, \quad \forall a \in A. \quad (1.1.40)$$

Де $\bar{y}_a$ – відомі поточні (на даний момент) значення пропускних спроможностей дуг. При обчисленні субградієнта функції $F_A(Y)$ внутрішню підзадачу потрібно розв'язувати $|T|$ раз.

Для розв'язання внутрішньої підзадачі (1.1.36) – (1.1.40) можна застосувати як схему декомпозиції за обмеженнями з використанням *r*-алгоритму, так і стандартні програми для розв'язування задач лінійного програмування (наприклад, методи внутрішніх точок [21, 22], що дозволяють врахувати блочну структуру підзадачі). При використанні *r*-алгоритму потрібна тільки підпрограма для знаходження найкоротших шляхів в орієнтованій мережі.

Щоб обчислити субградієнт функції $F_P(Y)$ потрібно $|T|$ раз розв'язати підзадачу знаходження двоїстих змінних до обмежень (1.1.42) для такої задачі лінійного програмування: знайти

$$\varphi_A^*(t) = \min_{x,y} \sum_{a\in A} Q_a y_{at} \quad (1.1.41)$$

при обмеженнях:

$$\sum_{k\in K} \sum_{p\in P_k} \delta_{kpa} z_{kpt} - y_{at} \leq \mu_{at}\left(y_a^0 + \bar{y}_a\right), \quad \forall a \in A, \quad (1.1.42)$$

$$\sum_{p\in P_k} z_{kpt} = d_k \quad \forall k \in K, \quad (1.1.43)$$

$$z_{kpt} \geq 0, \quad \forall k \in K,\ \forall p \in P, \quad (1.1.44)$$

$$y_{at} \geq 0, \quad \forall a \in A, \quad (1.1.45)$$

де $\bar{y}_a$ – відомі поточні значення пропускних спроможностей дуг, що доповнюють уже існуючі.

Підзадача (1.1.41) – (1.1.45) є простішою, ніж відповідна підзадача (1.1.36) – (1.1.40), і для її розв'язування можна використовувати r-алгоритм у комбінації зі схемою декомпозиції за обмеженнями (1.1.42).

1.1.6 Програмне забезпечення: програми SolverA та SolverP

Для розв'язання опуклих задач знаходження оптимальних пропускних спроможностей дуг для побудови відмовостійкої орієнтованої мережі реалізовані такі програми: програма **SolverA** для задачі (1.1.26) – (1.1.30) та програма **SolverP** для задачі (1.1.31) – (1.1.35) (мова програмування ФОРТРАН**)**. Алгоритми розв'язування обох задач базуються на подвійному використанні двох схем декомпозиції (одна в одній): схема декомпозиції за змінними, якими є невідомі значення доданих пропускних спроможностей дуг орієнтованої мережі; схема декомпозиції за обмеженнями (для розв'язання підзадач, які виникають для кожної одиничної відмови при фіксованих значеннях доданих пропускних спроможностей дуг). Відповідні вказаним схемам декомпозиції задачі негладкої оптимізації розв'язуються за допомогою r-алгоритму з адаптивним регулюванням крокового множника, де параметри r-алгоритму вибрані таким чином: коефіцієнт розтягу простору $\alpha = 4$, а параметри адаптивного регулювання крокового множника – $n_h = 3$, $q_1 = 1$, $q_2 = 1.1$. Максимальна кількість ітерацій, відведена r-алгоритму для розв'язування підзадач, вибрана рівною 500.

В результаті роботи програм SolverA та SolverP отримуємо такі параметри відмовостійкої орієнтованої мережі $N(V, A)$: мінімальні по сумарній вартості пропускні спроможності дуг мережі $N(V, A)$, що доповнюють вже існуючі, тобто для кожної дуги $a \in A$ знаходиться y_a^* – ресурс пропускної спроможності дуги a, що доповнює вже існуюче значення пропускної спроможності цієї дуги y_a^0; достатня умова того, що неможливо виконати усі вимоги на передачу об'ємів

потоків, щоб орієнтована мережа була відмовостійкою. Це може бути з двох причин: (a) структура мережі $N(V,A)$ така, що задовільнити вимоги до потоків неможливо; (b) структура одиничних відмов мережі $N(V,A)$ така, що задовільнити вимоги до потоків неможливо. Достатньою умовою невиконання усіх вимог на передачу об'ємів потоків в мережі є нерівність $\sum_{t\in T}\sum_{a\in A} Q_{at} y^*_{at} \gg 0$, яка має місце тоді і тільки тоді, коли в f^*_A або f^*_P вносить ненульовий вклад «штрафна» частина цільової функції.

RATFOR програму SolverA наведено в додатку А. Загальну структуру програми SolverA наведено на рисунку 1.1.5, де вказані блоки виконують такі функції.

Головна програма SolverA керує процесом знаходження розв'язку опуклої задачі А. В програмі встановлюються штрафні параметри (на основі вхідних даних), запускається процесс розв'язування задачі (1.1.26) – (1.1.30), аналізується отриманий розв'язок, результати розрахунку виводяться в файл протоколу роботи програми. Програма SolverA використовує підпрограми ReadDataA та CoordA.

Підпрограма ReadDataA читає вхідні дані задачі та перевіряє їх коректність за рядом ознак. Вхідні дані читаються послідовно з трьох файлів, які містять: (a) структуру орієнтованої мережі; (b) об'єми потоків між парами вершин, що пересилаються по мережі; (c) сценарій відмов у мережі.

Підпрограма CoordA реалізує роботу *r*-алгоритму для розв'язування координуючої негладкої задачі, пов'язаної зі схемою декомпозиції за змінними (використовує підпрограму FGCoord). Підпрограма FGCoord обчислює значення негладкої координуючої функції та її субградієнта, для чого послідовно для кожного з пошкоджень використовується підпрограма DualA.

Підпрограмма DualA знаходить двоїсті змінні до обмежень (1.1.37) підзадачі (1.1.36) – (1.1.40). Для їх знаходження за допомогою r -алгоритму розв'язується задача максимізації негладкої ввігнутої функції, що відповідає схемі декомпозиції за обмеженнями (1.1.37).

Підпрограма FGDual обчислює значення функції та її субградієнта для підпрограми DualA, для чого використовується метод знаходження найкоротшого шляху в орієнтованій використовується метод знаходження найкоротшого шляху в орієнтованій мережі [23, с. 137].

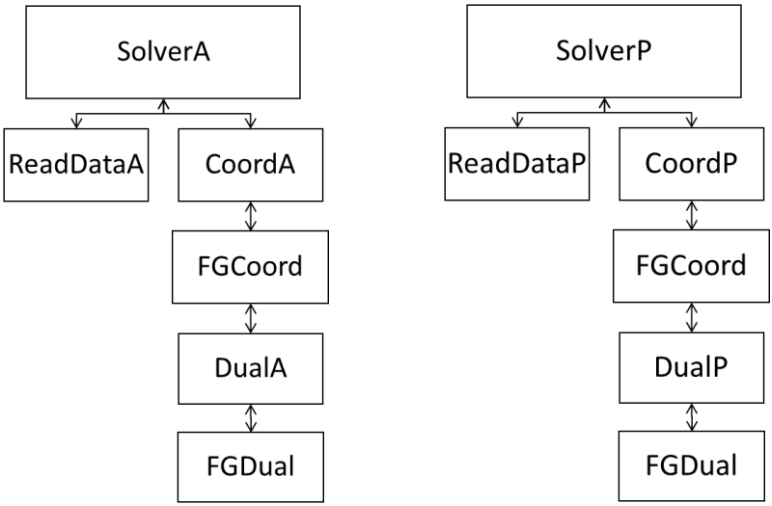


Рис. 1.1.5. Блок-схеми програм SolverA та SolverP

За аналогічною схемою, реалізовано програму SolverP (рисунок 1.1.5), де відповідні блоки виконують такі функції.

Головна програма SolverP керує процесом знаходження розв'язку опуклої задачі P. В програмі встановлюються штрафні параметри (на основі вхідних даних), запускається алгоритм розв'язування задачі (1.1.31) – (1.1.35), аналізується отриманий розв'язок, результати розрахунку виводяться в файл протоколу роботи програми. Програма SolverP використовує підпрограми ReadDataP та CoordP.

Підпрограма ReadDataP читає вхідні дані задачі та перевіряє їх коректність. Вхідні дані читаються послідовно з чотирьох файлів, які містять: (a) структуру орієнтованої мережі; (b) об'єми потоків, що

пересилаються по мережі; (c) структуру пошкоджень мережі; (d) допустимі шляхи для передачі потоків в мережі.

Підпрограма CoordP реалізує роботу *r*-алгоритму для розв'язування координуючої негладкої задачі, пов'язаної зі схемою декомпозиції за змінними *Y* для задачі (1.1.31) – (1.1.35). Підпрограма FGCoord обчислює значення негладкої координуючої функції та її субградієнта, які використовуються програмою CoordP. Для обчислення цих значень послідовно для кожного з пошкоджень використовується підпрограма DualC.

Підпрограмма DualC знаходить двоїсті змінні до обмежень (1.1.42) підзадачі (1.1.41) – (1.1.45) за допомогою *r*-алгоритму. Підпрограма FGDual обчислює значення функції та суперградієнта для підпрограми DualC.

Тестування та дослідження ефективності програм SolverA та SolverP проводилося для описаних вище лінійної функції $f_1(Y)$, квадратичної функції $f_2(Y)$ та нелінійної негладкої функції $f_3(Y)$ на прикладі орієнтованої мережі Net(6,16). Для порівняння було вибрано відому програму IPOPT [17] з NEOS-сервера. Результати тестування показали, що час розв'язання опуклих тестових задач з одиничними відмовами дуг за допомогою програми SolverA може бути суттєво менший за час розв'язання таких задач за допомогою програми IPOPT. Так, наприклад, для сценарію b) час розв'язання задачі A з функцією $f_1(Y)$ за допомогою програми SolverA дорівнює 2.5 сек., з функцією $f_2(Y)$ – 2.7 сек., з функцією $f_3(Y)$ – 2.6 сек., а за допомогою програми IPOPT відповідно 97 сек., 424 сек. та 99 сек. При цьому в усіх випадках було знайдено оптимальні розв`язки.

1.1.7 Висновки

Розглянуто математичні моделі двох класів задач знаходження пропускних спроможностей дуг відмовостійкої орієнтованої мережі. У першому класі задач (задача A) для передачі потоків можуть використовуватись всі можливі шляхи в мережі. У другому класі задач

(задача P) для передачі потоків задіяні тільки шляхи із наперед заданої множини шляхів. Математичні моделі представлені задачами лінійного, змішаного булевого лінійного та нелінійного програмування з блочною структурою матриці обмежень. Наведено програми SolverA та SolverP для розв'язання опуклих задач знаходження оптимальних пропускних спроможностей дуг для побудови відмовостійкої орієнтованої мережі. Результати обчислювальних експериментів з цими програмами демонструють їхню конкурентоспроможність як порівняти з програмою IPOPT.

Список літератури

1. Koren I., Krishna C.M. Fault tolerant systems. ELSEVIER 2020. 416 p.
2. Mostafa A. Design and Analysis of Reliable and Fault-Tolerant Computer systems, Kuwait 2006. 464p. https://doi.org/10.1142/p457
3. Nagurney A. Sustainable Transportation Networks, 2000. Edward Elgar Publishing Inc. 304 p.
4. Nagurney A., Qiang Q. Fragile Networks: Identifying Vunerabilities and Synergies in an Uncertain World. 2009. John W. Wiley & Sons. 344 p.
5. Шор Н.З., Сергієнко І.В., Шило В.П., Стецюк П.І. та ін. Задачі оптимального проектування надійних мереж. К.: Наук. Думка. 2005. 230 с.
6. Стецюк П.И., Жидков В.А. О двух задачах оптимизации пропускных способностей дуг отказоустойчивости сети. Материалы V-ой международной научной конференции «Транспортные системы и логистика». Кишинэу, 11–13 декабря 2013 г.. С. 300–309.
7. Stetsyuk P.I., Lykhovyd O.P. Two families of convex problems for finding capacities of arcs of a fault-tolerant network. Proceedings of 2019 IEEE 2nd Ukraine Conference on Electrical and Computer

Engineering (UKRCON), Lviv, Ukraine, July 2–6, 2019. P. 1057–1060. https://ieeexplore.ieee.org/Xplore/home.jsp
8. Сергиенко И.В., Стецюк П.И. Две ЛП-задачи с булевыми переменными для отказоустойчивой сети. *Інформатика та системні науки* (ICH-2014): матеріали V Всеукр. наук.-практ. конф. (м. Полтава, 13-15 березня 2014 року). Полтава: ПУЕТ, 2014. С. 284–287.
9. Стецюк П.І., Лиховид О.П., Омельяненко А.М. Задачі модернізації пропускних здатностей дуг відмовостійкої мережі. Тези доповідей XVIII міжнародної науково-практичної конференції «Математичне та програмне забезпечення інтелектуальних систем (MPZIS-2020)», м. Дніпро, 18–20 листопада 2020 р. Д.: ДНУ. 2020. С. 10–15.
10. Стецюк П.І., Лиховид О.П., Жидков В.О., Супрун А.А. Оптимізаційні задачі модернізації пропускних здатностей дуг відмовостійких мереж. Проблеми управління та інформатики. 2021. № 5. С. 5–20.
11. Gurobi Optimization, Inc., Gurobi Optimizer Reference Manual, 2014. https://www.gurobi.com
12. NEOS Solvers [e-resource]: https://neos-server.org/neos/solvers/
13. Fourer R., Gay D., Kernighan B. AMPL, A Modeling Language for Mathematical Programming. Belmont: Duxburry Press. 2003. 517 p.
14. Шор Н.З. Методы минимизации недифференцируемых функций и их приложения. К.: Наук. думка. 1979. 200 с.
15. Stetsyuk P.I. Shor's r-algorithms: theory and practice. In: Optimization Methods and Applications: In Honor of the 80th Birthday of Ivan V. Sergienko. Butenko S., Pardalos P.M, Shylo V. (Eds). Springer International Publishing. 2017. P. 495–520.
16. Стецюк П.И. Теория и программные реализации r-алгоритмов Шора. Кибернетика и системный анализ. 2017. Т. 53. № 5. С. 43–57.
17. IPOPT Solver [e-resource]: https://github.com/coin-or/Ipopt

1.2 БАГАТОПРОДУКТОВА ТРАНСПОРТНА ЗАДАЧА З УРАХУВАННЯМ МОДЕРНІЗАЦІЇ МЕРЕЖІ

М. Г. Журбенко

Анотація. Розглядається багатопродуктова транспортна задача з урахуванням модернізації мережі. Розроблено математичну модель, метод розв'язання і програмне забезпечення задач планування багатопродуктових потоків і модернізації транспортної мережі. Метод розв'язання базується на використанні алгоритмів негладкої оптимізації. Програмне забезпечення реалізовано в об'єктно-орієнтованому стилі мовою C++ і може використовуватися для розв'язання практичних задач великих розмірностей.

Abstract. A multi-product transportation problem taking into account the modernization of the network is considered. A mathematical model, solution method, and software for multi-product flow planning and transportation network modernization problems are developed. The solution method is based on the use of non-smooth optimization algorithms. The software is implemented in object-oriented style using C++ and can be used for solving practical problems of large dimensions.

1.2.1 Вступ

Математичні моделі планування багатопродуктових потоків на транспортних мережах мають різноманітні важливі практичні застосування (логістика, комунікаційні мережі). У даному підрозділі наводиться опис одного класу таких задач і методу їх розв'язання на основі використання алгоритмів негладкої оптимізації [1].

1.2.2 Математична модель

Модель описана на концептуальному рівні. Вхідні дані моделі позначаються словом data, вихідні – result.

1.2.3 Транспортна мережа

Транспортна мережа представляється орієнтованим графом $G = \{I, J\}$.

I (data) – множина вершин (вузлів, станцій) графа; i – індекс перерахування вершин. J (data) – множина орієнтованих дуг (ділянок) графа; j – індекс перерахування ребер. $J^-(i)\left(J^+(i)\right)$ (data) – множина ребер, що виходять з вершини (що входять в вершину); $i^-(j)\left(i^+(j)\right)$ (data) – початкова (кінцева) вершина ребра.

1.2.4 Кореспонденції перевезень

L ***(data***) – множина типів продуктів (вантажів); l – індекс перерахування типів продуктів. Q (data) – множина кореспонденцій; q – індекс перерахування кореспонденцій. Для кореспонденції визначені наступні дані: тип вантажу l_q (data); прогнозований обсяг перевезення b_q (data). У моделі буде використовуватися наступна варіантна схема реалізації кореспонденцій. Ця схема заснована на тому, що для кожної кореспонденції задано множина способів її реалізації Ω_q (data). Для кожної реалізації кореспонденції $\omega \in \Omega_q$ визначені наступні дані: $\Re_q(\omega)$ маршрут (data); тариф вартості перевезення одиниці вантажу $c_q^+(\omega)$ (data).

Маршрут $\Re_q(\omega)$ визначаться послідовним списком ділянок мережі і відповідає звичайному визначенню маршруту на орієнтованому графі:

$$\Re_q(\omega) = \left\{ j^1, j^2, ..., j^{n_q} \right\},$$

$$i^-\left(j^{k+1}\right)=i^+\left(j^k\right),\quad k=1,2,...,n_q-1,$$

де n_q – число ділянок маршруту $\Re_q(\omega)$.

Початкова станція $i_q^-(\omega)=i^-\left(j^1\right)$ маршруту $\Re_q(\omega)$ є однією зі станцій відправлення кореспонденції. Кінцева станція $i_q^+(\omega)=i^+\left(j^{n_q}\right)$ маршруту $\Re_q(\omega)$ є однією зі станцій призначення кореспонденції. Множинність станцій відправлення (призначення) може бути пов'язана, наприклад, з варіантністю відправлення вантажу з портів (доставкою вантажу в порти). Представлена форма визначення варіантів реалізацій кореспонденцій інформаційно надлишкова. Однак така форма забезпечує компактність запису і простоту інтерпретації моделі. Крім того вона має високу форму спільності і забезпечує облік різних варіантів постановки задачі. Формально множина варіантів реалізацій кореспонденції розглядається в моделі як вхідні дані. Як правило, ці дані програмно генеруються. При цьому в якості маршрутів кореспонденцій можуть вибиратися оптимальні шляхи за обраними критеріями (відстань; експлуатаційні витрати; час реалізації), що з'єднують станції відправлення і призначення кореспонденції. При генеруванні маршрутів кореспонденцій можуть враховуватися додаткові (технологічні, екологічні) вимоги, які в явному вигляді в моделі не відображені.

Кожній реалізації $\omega\in\Omega_q$ кореспонденції q відповідає змінна моделі $x_q^+(\omega)\geq 0$ (result) – обсяг перевезення кореспонденції q за варіантом її реалізації ω. X_q^+ – множина змінних для всіх кореспонденцій: $X_q^+=\left\{x_q^+ \mid q\in Q\right\}$.

Загальний обсяг реалізованих перевезень кореспонденції повинен бути не більшим, ніж заданий прогнозований максимальний обсяг кореспонденції b_q. Позначимо через $x_q^-\geq 0$ (result) нереалізований

обсяг кореспонденції q. c_q^- (data) – штрафний множник за наявність одиниці нереалізованого обсягу кореспонденції q. Зауважимо, що штрафні множники c_q^- відображають, як правило, не реальні фінансові втрати від реалізації не всього обсягу кореспонденції, а є керуючими параметрами моделі. Ці параметри дозволяють ранжувати кореспонденції за ступенем важливості реалізації їх перевезень. Крім того, введення змінних x_q^- забезпечує формальну спільність обмежень моделі.

Змінні $x_q^+(\omega)$ і x_q^- задовольняють балансові співвідношення:

$$\sum\left\{x_q^+(\omega) \mid \omega \in \Omega_q\right\} + x_q^- = b_q, \quad q \in Q,\ x_q^+(\omega) \geq 0,\ x_q^- \geq 0. \quad (1.2.1)$$

Тут X_q^- – множина всіх змінних x_q^-: $X_q^- = \left\{x_q^- \mid q \in Q\right\}$.

Загальний «дохід» $F^+\left(X_q^+\right)$ від реалізацій кореспонденцій і загальний «штраф» $F_b^-\left(X_q^-\right)$ за наявність нереалізованих перевезень визначаються такими рівностями:

$$F^+\left(X_q^+\right) - \sum\sum\left\{c_q^+(\omega)x_q^+(\omega) \mid \omega \in \Omega_q;\ q \in Q\right\},$$

$$F_b^-\left(X_q^-\right) - \sum\left\{c_q^+ x_q^- \mid q \in Q\right\}.$$

Тут $F_c^-\left(X_q^+\right)$ (result) – експлуатаційні витрати на перевезення всіх кореспонденцій: $F_c^-\left(X_q^+\right) = \sum\left\{c_{qj}^- w^q(j) \mid q \in Q;\ j \in J\right\}$, де c_{qj}^- (data) – витрати на перевезення одиниці об'єму кореспонденції q на ділянці j.

1.2.5 Пропускна спроможність ділянок транспортної мережі

J^* (data) – множина ділянок транспортної мережі з обмеженою пропускною спроможністю $\left(J^* \subset J\right)$. r_j (data) – пропускна спроможність ділянки j. Нехай $w^q(j)$ (result) – обсяг кореспонденції, який перевозиться по ділянці j. Значення $w^q(j)$ однозначно визначаються значеннями змінних $x_q(\omega)$:

$$w^q(j) = \sum\left\{x_q^+(\omega) \mid j \in \Re_q(\omega);\ \omega \in \Omega_q\right\}.$$

Тоді умови обмеженості пропускної спроможності ділянок з урахуванням їх реконструкції відповідають наступним обмеженням моделі:

$$\sum\left\{\gamma_q w^q(j) \mid q \in Q\right\} \le r_j + x_j^r,\ \ j \in J^*, \tag{1.2.2}$$

$$x_j^r \ge 0,\ \ j \in J^*, \tag{1.2.3}$$

де γ_q (date) – масштабуючий коефіцієнт, який визначається типом кореспонденції q, x_j^r (result) – приріст потужності пропускної спроможності ділянки за рахунок його реконструкції.

X_r – множина всіх змінних x_j^r:

$$X_r = \left\{x_j^r \mid j \in J^*\right\}.$$

$F_r^-(X_r)$ – загальний обсяг фінансів реконструкції ділянок транспортної мережі:

$$F_r^-(X_r) = \sum\left\{c_j^r x_j^r \mid j \in J^*\right\},$$

де c_j^r (date) – питомі фінансові витрати реконструкції ділянки (витрати при збільшенні пропускної спроможності на одну одиницю).

B^f (data) – максимальний обсяг фінансів реконструкції ділянок транспортної мережі:

$$F_r^-\left(X_r\right) \le B^f . \qquad (1.2.4)$$

Задача полягає у максимізації функції «загального доходу» $F\left(X_q^+, X_q^-, X_r\right)$:

$$\max \leftarrow F\left(X_q^+, X_q^-, X_r\right) =$$

$$= F^+\left(X_q^+\right) - F_b^-\left(X_q^-\right) - F_c^-\left(X_q^+\right) - F_r^-\left(X_r\right) \qquad (1.2.5)$$

за обмежень (1.2.1), (1.2.2), (1.2.3).

Змістовний сенс функції $F\left(X_q^+, X_q^-, X_r\right)$: {оплата за доставку} – {штраф за недопоставлені обсяги} – {експлуатаційні витрати} – {витрати на реконструкцію}.

Основними змінними задачі є наступні змінні моделі: X_q^+ (обсяги реалізованих перевезень кореспонденцій), X_q^- (нереалізовані обсяги перевезень кореспонденцій), X_r (змінні збільшення пропускних спроможностей ділянок). Решта змінних і зазначені вище співвідношення є допоміжними і введені для компактності запису і простоти інтерпретації моделі.

1.2.6 Метод розв'язання

Задача (1.2.1) – (1.2.5) представлена в формі задачі лінійного програмування. Основною особливістю реальних задач розглянутого класу є їх велика розмірність: число кореспонденцій $\left(|Q|\right)$ і ділянок мережі $\left(|J|\right)$ можуть досягати сотень тисяч. Множина обмежень задачі характеризується яскраво вираженою блочною структурою. «Зв'язуючими» є обмеження по пропускній спроможності окремих ділянок мережі (1.2.2) і обмеження (1.2.4). Число обмежень (1.2.2), як правило, значно менше ніж число всіх ділянок мережі і складає порядку декількох сотень. Зазначені особливості моделі обумовлюють

недоцільність використання стандартного програмного забезпечення для її чисельної реалізації.

Метод розв'язання задачі (1.2.1) – (1.2.5) заснований на використанні алгоритмів негладкої оптимізації в схемах декомпозиції блокових задач математичного програмування по зв'язуючих обмеженнях [2] і складається з двох етапів.

На першому етапі розв'язується двоїста до (1.2.1) – (1.2.5) задача відносно обмежень (1.2.2), (1.2.4):

$$\psi\left(U^r, u^f\right) \to \max, \tag{1.2.6}$$

$$u_j^r \geq 0, \;\; j \in J^*; \; u^f \geq 0, \tag{1.2.7}$$

u_j^r – двоїста змінні, відповідні обмеженням (1.1.2),

$U^r = \left\{u_j^r \mid j \in J^*\right\}$;

u^f – двоїста змінна, відповідна обмеженню (1.1.4),

$\psi\left(U^r, u^f\right) = \min\left\{L\left(X_q^+, X_q^-, X_r, U^r, u^f\right) \mid X_q^+, X_q^-, X_r\right\}$,

$L\left(X_q^+, X_q^-, X_r, U^r, u^f\right)$ – функція Лагранжа задачі (1.2.1) – (1.2.5) щодо обмежень (1.2.2), (1.2.4).

$$L\left(X_q^+, X_q^-, X_r, U^r, u^f\right) = F\left(X_q^+, X_q^-, X_r\right) - \sum u_j^r \left\{\sum\left\{\gamma_q w^q(j) \mid q \in Q\right\} - r_j - x_j^r \mid j \in J^*\right\}$$

Мінімізація функції Лагранжа по змінним X_q^+, X_q^-, X_r виконується при фіксованих значеннях двоїстих змінних X_q^+, X_q^-, X_r з урахуванням обмежень (1.2.1), (1.2.3).

В силу блокової структури обмежень (1.2.1), (1.2.3) і сепарабельності цільової функції (1.2.5) задача мінімізації функції Лагранжа зводиться до незалежного розв'язання багатьох найпростіших підзадач: $|Q|$ підзадач від $|\Omega_q|+1$ змінних $\left(x_q^+(\omega), x_q^-\right)$ і $\left|J^*\right|$ підзадач від однієї змінної $\left(x_j^r\right)$. Власне в цьому і полягає сенс декомпозиційного методу. Для лінійних залежностей розв'язки

зазначених підзадач визначаються аналітично, для нелінійних залежностей розв'язання зводиться до використання мало витратного чисельного алгоритму.

Наведемо розв'язання для першої групи підзадач по змінним $\left(x_q^+(\omega), x_q^-\right)$. Кожна підзадача цієї групи (для кожної кореспонденції q) полягає в наступному.

$$\max\left\{\sum\left\{\tilde{c}_q^+(\omega)x_q^+(\omega)\,|\,\omega\in\Omega_q\right\}-c_q^-x_q^-\right\}, \qquad (1.2.8)$$

$$\sum\left\{x_q^+(\omega)\,|\,\omega\in\Omega_q\right\}+x_q^-=b_q,\ \ x_q^+(\omega)\geq 0,\ \ x_q^-\geq 0, \qquad (1.2.9)$$

де коефіцієнти $\tilde{c}_q^+(\omega)$ («збурені» коефіцієнти $c_q^+(\omega)$ з урахуванням значень двоїстих змінних u_j^r) визначаються за формулою:

$$\tilde{c}_q^+(\omega)=c_q^+(\omega)-\gamma_q\sum\left\{u_j^r\,|\,j\in\Re_q(\omega)\right\}.$$

Розв'язання задач (1.2.8) – (1.2.9) $x_q^+\left(\omega;U^r\right)$, $x_q^-\left(U^r\right)$ визначається наступним чином. Нехай ω^* реалізація кореспонденції, для якої значення $\tilde{c}_q^+(\omega)$ максимально (якщо таких реалізацій кілька, вибирається будь-яка з них):

$$\tilde{c}_q^+\left(\omega^*\right)=\max\left\{\tilde{c}_q^+(\omega)\,|\,\omega\in\Omega_q\right\}. \qquad (1.2.10)$$

Якщо $\tilde{c}_q^+\left(\omega^*\right)\geq -c_q^+$, то $x_q^+\left(\omega^*;U^r\right)=b_q$, $x_q^+\left(\omega;U^r\right)=0$ для $\omega\neq\omega^*$ і $x_q^-\left(U^r\right)=0$. Якщо $\tilde{c}_q^+\left(\omega^*\right)<-c_q^+$, то $x_q^+\left(\omega;U^r\right)=0$ для всіх $\omega\in\Omega_q$ і $x_q^-\left(U^r\right)=b_q$.

Підзадача другої групи (по змінним x_j^r) для фіксованого значення $j\in J^*$ полягає в наступному.

$$\max\tilde{c}_j^r x_j^r, \qquad (1.2.11)$$

$$\bar{x}_j^r \geq x_j^r \geq 0\,, \qquad (1.2.12)$$

де коефіцієнти $\tilde{c}_j^r$ («збурені» коефіцієнти c_j^r з урахуванням значень двоїстих змінних) визначаються за формулою $\tilde{c}_j^r = -c_j^r + u_j^r + u_f$.

Розв'язки задачі (1.2.11) – (1.2.12) $x_j^r\left(U^r\right)$:

$$x_j^r\left(U^r\right) = \begin{cases} 0, & c_j^r \leq 0, \\ \bar{x}_j, & \tilde{c}_j^r \geq 0. \end{cases}$$

Обчислення субградієнтів функції $\psi\left(U^r, u^f\right)$ зводиться до обчислення нев'язок обмежень (1.2.2), (1.2.4) для отриманих при розв'язанні підзадач значеннях змінних [2].

Для розв'язання двоїстої задачі використовується нова модифікація r-алгоритму [3] – алгоритму мінімізації з використанням операції розтягу простору в напрямку різниці двох послідовних субградієнтів. На відміну від r -алгоритму [4], значення коефіцієнтів розтягу в запропонованій модифікації розраховуються в процесі роботи алгоритму. Алгоритм може використовуватися з постійним кроком.

Другий етап розв'язання задачі полягає у визначенні оптимальних значень змінних вихідної задачі. Є різні варіанти розв'язання цієї проблеми [2, 5]. Відзначимо, що для більшості прямих змінних їх оптимальні значення визначаються безпосередньо на підставі розв'язання двоїстої задачі [6]. Після виключення цих змінних і несуттєвих обмежень (для таких обмежень відповідні отримані оптимальні значення рівні нулю) з вихідної задачі ми отримуємо задачу істотно меншої розмірності. Є такі варіанти розв'язання цієї задачі: використовувати симплекс алгоритм; застосувати прийом малого квадратичного збурення цільової функції (з метою забезпечення її строгої опуклості); використовування прийому усереднення значень змінних в процесі субградієнтного алгоритму розв'язання двоїстої задачі [7].

Відзначимо, що трудомісткість розв'язання задачі в основному (≈ 90 %) визначається розв'язанням двоїстої задачі (1.2.6) – (1.2.7).

1.2.7 Програмне забезпечення

Програмне забезпечення математичної моделі розроблено на мові С++ в стилі об'єктно-орієнтованого програмування. Всі змістовні об'єкти задачі (станція, ділянка, маршрут, кореспонденція та інші) відображені відповідними класами С ++.

Задача (1.2.1) – (1.2.5) представлена в формі задачі лінійного програмування. Однак при розробці її програмного забезпечення передбачена можливість врахування нелінійності окремих функціоналів моделі (наприклад, експлуатаційних витрат реалізації перевезення від її обсягу).

Як базові елементи програмного забезпечення використовують такі алгоритми:

- алгоритм знаходження найкоротших шляхів на графі;
- алгоритм розв'язання транспортної задачі на графі;
- алгоритм негладкої оптимізації.

Вхідні дані визначають наступну інформацію:

- транспортна мережа (орієнтований граф транспортної мережі, станції, ділянки);
- види агрегованих перевезень (множина різних типів вантажів);
- прогнозований обсяг перевезень кореспонденцій;
- тарифи провізних плат і експлуатаційних витрат при реалізації перевезень.
- обсяг капітальних фінансових коштів, призначених для реконструкції ділянок (нарощування пропускної спроможності ділянок).

Вихідні дані програмного забезпечення визначають чисельну інформацію з основних питань розглянутого класу задач перспективного планування:

- раціональну схему перевезень (реалізацію кореспонденцій) з урахуванням отримання максимального доходу;

- критичні по пропускній спроможності ділянки транспортної мережі;
- рекомендації щодо раціонального розподілу фінансових коштів на реконструкцію критичних ділянок транспортної мережі.

Вихідні дані можуть бути використані для прийняття рішень перспективного розвитку транспортної системи та об'єктивного обґрунтування по її реконструкції.

1.2.8 Висновки

Розроблена математична модель призначена для вирішення перспективного планування функціонування та модернізації транспортної системи. Транспортна система зображується як орієнтований граф. Вершини графа відповідають пунктам (станціям) постачальників/споживачів вантажів різного типу. Ребра графа відповідають ділянкам транспортної системи, якими здійснюється перевезення вантажів. Для ділянок мережі визначається їхня пропускна спроможність за обсягами реалізації перевезень.

Основна особливість представленої моделі полягає у використанні варіантів реалізації кореспонденцій перевезень. Формально множина варіантів реалізації кореспонденції розглядається в моделі як вхідні дані. Як правило, ці дані програмно генеруються. При цьому як маршрути кореспонденцій можуть вибиратися оптимальні шляхи за обраними критеріями (відстань, експлуатаційні витрати, час реалізації), що з'єднують станції відправлення та призначення кореспонденції.

Практичні задачі даного класу характеризуються великою розмірністю відповідних задач математичного програмування. Мережевий характер обмежень цих задач дозволяє розробити ефективні методи їх реалізації, враховуючи ці особливості. Запропонований метод для розв’язання розробленої моделі базується на використанні алгоритмів негладкої оптимізації в поєднанні зі схемою декомпозиції по зв'язуючих обмеженнях задачі [2].

Список літератури

1. Журбенко М.Г., Чумаков Б.М. До задачі планування багатопродуктових потоків і модернізації транспортної мережі. *Кібернетика та комп'ютерні технології*. 2021. № 4. С. 5–11.
2. Шор Н.З. Методы минимизации недифференцируемых функций и их приложения. К.: Наук. думка. 1979. 200 с.
3. Журбенко Н.Г., Лиховид А.П. К численной эффективности одной модификации r-алгоритма. *Комп'ютерна математика*. К.: Ин-т кибернетики им. В.М. Глушкова НАН Украины. 2019. № 1. С. 2–10.
4. Шор Н.З., Журбенко Н.Г. Метод минимизации, использующий операцию растяжения пространства в направлении разности двух последовательных градиентов. Кибернетика. Киев: Наук. думка. 1971. № 3. С. 51–59.
5. Белых Т.В., Журбенко Н.Г., Шулинок Э.И. Динамические производственно-распределительные задачи. Теорія оптимальних рішень. К.: Ін-т кібернетики ім. В.М. Глушкова НАН України. 2019. № 18. С. 61–66.
6. Журбенко Н.Г. К двухэтапной схеме декомпозиционного метода решения блочных задач линейного программирования. *Теория оптимальных решений*. Киев: Ин-т кибернетики им. В.М. Глушкова НАН Украины. 2001. С. 22–26.
7. Беляева Л.В., Шор Н.З., Журбенко Н.Г. О методе решения одного класса динамических распределительных задач. *Экономика и математические методы*. 1978. № 14(1). С. 137–146.

1.3 ЗАДАЧІ ПРО НАЙКОРОТШІ K-ВЕРШИННІ ЦИКЛИ ТА ШЛЯХИ

П. І. Стецюк, О. О. Жмуд

Анотація. Розглядаються математичні моделі для задач про найкоротші цикли та шляхи, які проходять через задану кількість вершин орієнтованого графа. Для пошуку найкоротшого циклу сформульовано дві задачі – задача змішаного булевого і лінійного програмування та задача дискретного програмування. Досліджено ефективність розв'язання задач за допомогою сучасних версій Gurobi та CPLEX. Для пошуку найкоротшого шляху побудовано задачу змішаного булевого і лінійного програмування. За її допомогою знайдено оптимальні маршрути для відвідування пунктів виноробства Малопольського винного шляху у напрямку Львів-Вроцлав-Львів.

Abstract. Mathematical models for problems of shortest cycles and paths passing through a given number of vertices of a directed graph are considered. To find the shortest cycle, two problems are formulated – a mixed Boolean and linear programming problem and a discrete programming problem. The effectiveness of solving problems using modern versions of Gurobi and CPLEX was investigated. A mixed Boolean and linear programming problem was constructed to find the shortest path. With its help, optimal routes were found for visiting the points of wine production of the Malopolska Wine Route in the direction of Lviv – Wrocław – Lviv.

1.3.1 Вступ

При розв'язанні конкретних транспортних та логістичних проблем в умовах, що постійно змінюються, часто виникає необхідність пошуку найкоротшого шляху. «Задачі про найкоротший шлях, в яких на шляхи накладаються деякі обмеження, часто настільки складні, що відповідні алгоритми дозволяють знаходити оптимальні

розв'язку лише таких задач, розміри яких (наприклад, число вершин) на кілька порядків менше, ніж у аналогічних задач без обмежень» [1, с. 177]. До задач про найкоротші шляхи з обмеженнями відносяться NP-важкі задачі – задача комівояжера (найкоротший гамільтонів цикл) і задача пошуку найкоротшого гамільтонового шляху.

У цьому підрозділі розглянуто оптимізаційні задачі знаходження найкоротших циклів і шляхів, що проходять через задану кількість вершин орієнтованого графа [2]. Сформульовано дві задачі математичного програмування для знаходження найкоротшого циклу. Щоб забезпечити зв'язність цикл у використано ідею моделювання задачі про потік [3] та використано обмеження, аналогічні побудованим для задачі комівояжера [4]. Встановлено зв'язок формулювань обох задач із задачею комівояжера і досліджено ефективність їх розв’язання за допомогою сучасних версій Gurobi і CPLEX. Наведено формулювання задачі про найкоротший k-вершинний шлях. З її допомогою знайдено оптимальні маршрути для відвідування пунктів виноробства Малопольського винного шляху в напрямку Львів – Вроцлав – Львів.

1.3.2 Два формулювання задачі про найкоротший k-вершинний цикл

Нехай $D_{n,n}$ – повний граф, де n – кількість вершин, а $d_{ij} > 0$ – довжина дуги між вершинами i і j, $i \neq j$. Зафіксуємо вершину s в графі $D_{n,n}$. Цикл, який починається і закінчується в вершині s і проходить через k вершин, де $1 \le k \le n-1$ (вершина s в розрахунок не береться), будемо називати k-вершинним циклом в графі $D_{n,n}$. Якщо $k = n-1$, то цей цикл збігається з гамільтоновим циклом, який проходить через всі вершини графа $D_{n,n}$. k-Вершинний цикл, якому відповідає найменша сумарна довжина $(k+1)$ дуг, що входять до

нього, будемо називати найкоротшим k-вершинним циклом, а його довжину позначимо d_k^*.

Розглянемо два формулювання задач цілочислового лінійного програмування для знаходження найкоротшого k-вершинного циклу. Щоб забезпечити зв'язність циклу в першому формулюванні використовується ідея моделювання задачі про потік [3], а в другому використовуються обмеження, аналогічні тим, які побудовані С. Міллером, А. Таккером та Р. Земліним для задачі комівояжера [4]. З обох формулювань випливають відомі постановки задач комівояжера [5, 6].

Перше формулювання. Нехай x_{ij} – булева змінна, що дорівнює одиниці, якщо в цикл входить дуга, що з'єднує вершини i та j, і нулю в іншому випадку. Так як $i \neq j$, то кількість таких змінних дорівнює $n(n-1)$. Нехай булева змінна y_i дорівнює одиниці, якщо цикл проходить через вершину i, і нулю в іншому випадку. Кількість таких змінних дорівнює $(n-1)$, тому що вершина s зафіксована. Позначимо z_{ij} невід'ємну змінну, яка задає величину потоку певного продукту від вершини i до вершини j. Цих змінних рівно стільки ж, як і булевих змінних x_{ij}.

Знаходженню найкоротшого k-вершинного циклу відповідає задача змішаного булевого і лінійного програмування [7, 8]:

знайти

$$d_k^* = \min_{x_{ij}} \sum_{i=1}^{n} \sum_{j=1,\, j\neq i}^{n} d_{ij} x_{ij} \qquad (1.3.1)$$

за обмежень

$$\sum_{j=1,\, j\neq s}^{n} x_{sj} = 1, \quad \sum_{j=1,\, j\neq i}^{n} x_{ij} = y_i, \quad i = 1,\ldots,n, \quad i \neq s, \qquad (1.3.2)$$

$$\sum_{j=1,\, j\neq s}^{n} x_{js} = 1, \quad \sum_{j=1,\, j\neq i}^{n} x_{ji} = y_i, \quad i = 1,\ldots,n, \quad i \neq s, \qquad (1.3.3)$$

$$\sum_{i=1, i\neq s}^{n} y_i = k\,, \tag{1.3.4}$$

$$z_{ij} - kx_{ij} \le 0, \quad i, j = 1,...,n, \quad i \neq j\,, \tag{1.3.5}$$

$$\sum_{j=1, j\neq s}^{n} z_{sj} = k, \qquad \sum_{j=1, j\neq s}^{n} z_{js} = 0\,, \tag{1.3.6}$$

$$\sum_{j=1, j\neq i}^{n} z_{ij} - \sum_{j=1, j\neq i}^{n} z_{ji} = -y_i, \quad i = 1,\ldots,n, \quad i \neq s\,, \tag{1.3.7}$$

$$x_{ij} = 0 \vee 1, \quad i, j = 1,...,n, \quad i \neq j\,, \tag{1.3.8}$$

$$y_i = 0 \vee 1, \quad i = 1,\ldots,n, \quad i \neq s\,, \tag{1.3.9}$$

$$z_{ij} \ge 0, \quad i, j = 1,...,n, \quad i \neq j\,. \tag{1.3.10}$$

Теорема 1.3.1 [7]. *Якщо* k – ціле число, яке задовольняє нерівності $1 \le k \le n-1$, то обмеження (1.3.2) – (1.3.10) описують всі можливі k-вершинні цикли цикли, які починаються і закінчуються в вершині s графа $D_{n,n}$.

Мінімізація цільової лінійної функції в (1.3.1) відповідає знаходженню k-вершинного циклу мінімальної довжини d_k^*. Обмеження (1.3.8) визначають булеві змінні x_{ij}, обмеження (1.3.9) – булеві змінні y_i, а обмеження (1.3.10) – невід'ємні змінні z_{ij}. Умови (1.3.2) – (1.3.7) мають наступний зміст.

Обмеження (1.3.2) описують одноразовий вхід в вершину s і в ті k вершин, для яких $y_i = 1$, а обмеження (1.3.3) описують одноразовий вихід з вершини s і тих k вершин, для яких $y_i = 1$. Обмеження (1.3.4) задає умову, що рівно для k вершин змінні $y_i = 1$, і визначає набір вершин, через які проходить цикл, що починається в вершині s.

Сімейства обмежень (1.3.5), (1.3.6) і (1.3.7) гарантують зв'язність k-вершинного циклу. Обмеження (1.3.5) забезпечують перевезення продукту між вершинами i і j тільки в тому випадку, якщо $x_{ij} = 1$.

Обмеження (1.3.6), (1.3.7) означають, що з вершини s необхідно вивезти k одиниць продукту, залишаючи в кожній з вершин циклу лише одну одиницю продукту. Це дозволяє уникнути підциклів в графі $D_{n,n}$ і забезпечує зв'язність k-вершинного циклу мінімальної довжини d_k^*.

Задача $(1.3.1)-(1.3.10)$ містить $N_1 = 2n^2 - n - 1$ змінних, з яких $n^2 - 1$ є булеві, а $n(n-1)$ – невід'ємні, і $M_1 = n^2 + 2n + 2$ обмежень, у тому числі $3n+2$ – лінійні рівності, а $n(n-1)$ – лінійні нерівності.

Відзначимо, що формулювання задачі $(1.3.1)-(1.3.10)$ може бути застосоване і для неповного графа, якщо його доповнити відсутніми дугами і значення довжин для них встановити рівними сумі довжин всіх дуг неповного графа.

Друге формулювання. Тут булеві змінні x_{ij} і y_i залишаються такими ж, як і в першому формулюванні. Зв'язність k-вершинного циклу в графі $D_{n,n}$ забезпечуватимуть нові цілочислові змінні u_i, значення яких відповідають номеру кроку, на якому відвідується вершина i. Оскільки вершину s відвідувати не потрібно, то кількість змінних u_i дорівнюватиме $(n-1)$.

Знаходженню найкоротшого k-вершинного циклу циклу відповідає наступна задача цілочислового лінійного програмування [8]:

знайти

$$d_k^* = \min_{x_{ij}} \sum_{i=1}^{n} \sum_{j=1,\, j\neq i}^{n} d_{ij} x_{ij} \qquad (1.3.11)$$

за обмежень

$$\sum_{j=1,\, j\neq s}^{n} x_{sj} = 1, \quad \sum_{j=1,\, j\neq i}^{n} x_{ij} = y_i, \quad i = 1,\ldots,n, \quad i \neq s, \qquad (1.3.12)$$

$$\sum_{j=1,\, j\neq s}^{n} x_{js} = 1, \quad \sum_{j=1,\, j\neq i}^{n} x_{ji} = y_i, \quad i = 1,\ldots,n, \quad i \neq s, \qquad (1.3.13)$$

$$\sum_{i=1, i\neq s}^{n} y_i = k, \tag{1.3.14}$$

$$u_i - u_j + kx_{ij} \le k-1, \quad i, j = 1,...,n, \quad i \neq s,\ j \neq s,\ i \neq j, \tag{1.3.15}$$

$$x_{ij} = 0 \vee 1, \quad i, j = 1,...,n, \quad i \neq j, \tag{1.3.16}$$

$$y_i = 0 \vee 1, \quad i = 1,\ldots,n, \quad i \neq s, \tag{1.3.17}$$

$$u_i - \text{цілі числа}, \quad 1 \le u_i \le k, \quad i = 1,\ldots,n, \ i \neq s. \tag{1.3.18}$$

Теорема 1.3.2 [8]. Якщо k – ціле число, яке задовольняє нерівностям $1 \le k \le n-1$, то обмеження (1.3.12) – (1.3.18) описують всі можливі k-вершинні цикли, які починаються і закінчуються в вершині s графа $D_{n,n}$.

Мінімізація лінійної функції в (1.3.11) відповідає знаходженню k-вершинного циклу мінімальної довжини d_k^*, а обмеження (1.3.12) – (1.3.14) аналогічні обмеженням (1.3.2) – (1.3.4) і визначають ті k вершин в графі $D_{n,n}$, через які проходить цикл, що починається в вершині s. Обмеження (1.3.15) забезпечують зв'язність k-вершинного циклу та визначають порядок відвідуваності вершин циклу.

Припустимо, що є два цикли. Один з них не проходить через вершину s. Позначимо його $(i_1,\ldots,i_p,i_1)$. З обмежень (1.3.15) випливає, що для кожної пари вершин, що йдуть по порядку в цьому циклі, справедливі наступні нерівності:

$$u_{i_1} - u_{i_2} + k \le k-1,$$
$$u_{i_2} - u_{i_3} + k \le k-1,$$
$$\vdots$$
$$u_{i_p} - u_{i_1} + k \le k-1.$$

Склавши ці нерівності, отримаємо $pk \le p(k-1)$, що неможливо при $p \neq 0$. Отже, для будь-якого підцикла, що не проходить через вершину s, обмеження (1.3.15) не виконуються.

Переконаємося, що цикл, що проходить через k вершин, для яких $y_i = 1$, задовольняє обмеженням (1.3.15), тобто можна підібрати відповідні значення змінних u_i. Нехай $u_i = p$, якщо вершина i, для якої $y_i = 1$, відвідується на кроці p. Тоді нерівність $u_i - u_j \le k - 1$ виконується при $x_{ij} = 0$, так як $p \le k$, а змінна $u_j \ge 1$. Якщо $x_{ij} = 1$, то $u_i = p$, а $u_j = p + 1$. В цьому випадку $p - (p+1) + k \le k - 1$, звідси, $k - 1 \le k - 1$ і обмеження виду (1.3.15) виконується як строга рівність.

Задача (1.3.11) – (1.3.18) містить $N_2 = n^2 + n - 2$ змінних, з яких $n^2 - 1$ є булеві, а $(n-1)$ – цілочислові, і $M_2 = n^2 - n + 3$ обмежень, у тому числі $2n+1$ – лінійні рівності, а $(n-1)(n-2)$ – лінійні нерівності. Задача (1.3.11) – (1.3.18) може бути застосована для неповного графа, якщо незв'язані між собою вершини графа з'єднати дугами, приписавши їм значення довжин, що дорівнюють сумі довжин всіх дуг неповного графа.

1.3.3 Задача комівояжера і обчислювальні експерименти

Якщо $k = n - 1$, то з обмеження (1.1.4) усі булеві змінні y_i дорівнюють одиниці і $(n-1)$-вершинний цикл збігається з гамільтоновим циклом. Знаходженню найкоротшого гамільтонового циклу відповідає задача змішаного булевого і лінійного програмування:

знайти

$$d_k^* = \min_{x_{ij}} \sum_{i=1}^{n} \sum_{j=1, j \ne i}^{n} d_{ij} x_{ij} \qquad (1.3.19)$$

за обмежень

$$\sum_{j=1, j \ne i}^{n} x_{ij} = 1, \quad \sum_{j=1, j \ne i}^{n} x_{ji} = 1, \quad i = 1, ..., n, \qquad (1.3.20)$$

$$z_{ij} - (n-1) x_{ij} \le 0, \quad i, j = 1, ..., n, \quad i \ne j, \qquad (1.3.21)$$

$$\sum_{j=1,\, j\neq s}^{n} z_{sj} = n-1, \quad \sum_{j=1,\, j\neq s}^{n} z_{js} = 0, \tag{1.3.22}$$

$$\sum_{j=1,\, j\neq i}^{n} z_{ij} - \sum_{j=1,\, j\neq i}^{n} z_{ji} = -1, \quad i = 1,\ldots,n, \quad i \neq s, \tag{1.3.23}$$

$$x_{ij} = 0 \vee 1, \quad z_{ij} \geq 0, \quad i, j = 1,...,n, \quad i \neq j. \tag{1.3.24}$$

Задача (1.3.19) – (1.3.24) збігається з формулюванням задачі комівояжера, наведеної в [8, ст. 46]. Обчислювальні аспекти щодо її розв'язання за допомогою сучасних програм Gurobi і CPLEX можна знайти в [9–12].

Якщо $k = n-1$, то для знаходження найкоротшого гамільтонового циклу з другого формулювання (1.3.11) – (1.3.18) випливає задача цілочислового лінійного програмування:

знайти

$$d_k^* = \min_{x_{ij}} \sum_{i=1}^{n} \sum_{j=1,\, j\neq i}^{n} d_{ij} x_{ij} \tag{1.3.25}$$

за обмежень

$$\sum_{j=1,\, j\neq i}^{n} x_{ij} = 1, \quad \sum_{j=1,\, j\neq i}^{n} x_{ji} = 1, \quad i = 1,...,n, \tag{1.3.26}$$

$$u_i - u_j + (n-1)x_{ij} \leq n-2, \quad i, j = 1,...,n, \quad i \neq s,\ j \neq s, i \neq j, \tag{1.3.27}$$

$$x_{ij} = 0 \vee 1, \quad i, j = 1,...,n, \quad i \neq j. \tag{1.3.28}$$

$$u_i - \text{цілі числа}, 1 \leq u_i \leq (n-1), \quad i = 1,\ldots,n, \quad i \neq s. \tag{1.3.29}$$

Задача (1.3.25) – (1.3.29) відрізняється від формулювань задачі комівояжера, наведених в [6, ст. 45] і [7, ст. 65], де зв'язність маршруту, що проходить через n міст (вершин), забезпечується на основі умов:

$$u_i - u_j + nx_{ij} \leq n-1, \quad i, j = 1,...,n, \quad i \neq j. \tag{1.3.30}$$

Обмеження (1.3.30) використовують на одну вершину більше, так як вони не вимагають, щоб цикл проходив через вершину s.

Використання як задачі (1.3.1) – (1.3.10), так і задачі (1.3.11) – (1.3.18) розширює можливості застосувань в порівнянні з задачами комівояжера (1.3.19) – (1.3.24) і (1.3.25) – (1.3.29). Так, наприклад, задачі (1.3.1) – (1.3.10) і (1.3.11) – (1.3.18) можуть бути легко адаптовані для визначення кільцевих маршрутів при плануванні пасажирських і вантажних перевезень [11]. І якщо за допомогою обмежень вигляду (1.3.5) – (1.3.7) легко врахувати транспортування заданих обсягів вантажу, то за допомогою обмежень вигляду (1.3.15) легко врахувати порядок відвідування деяких пунктів призначення.

Для знаходження оптимальних розв'язків задач (1.3.1) – (1.3.10) і (1.3.11) – (1.3.18) можна використовувати сучасне програмне забезпечення для вирішення завдань змішаного цілочислового і лінійного програмування. Найбільш популярними з таких програм є програми Gurobi [13] і CPLEX [14], їх сучасні версії Gurobi 9.1.1 і CPLEX 20.1.0.0 доступні на NEOS-сервері [15].

Щоб дослідити в якому співвідношенні знаходяться «час роботи зазначених програм» і «розміри задач (графів)» на мові моделювання AMPL [16] було розроблено опис математичних моделей для задач (1.3.1) – (1.3.10) і (1.3.11) – (1.3.18) і адаптовано для графів формату *.tsp, де вершини NODES ототожнюються з координатами точок на площині – xcoord {NODES} та ycoord {NODES}, а відстані між кожною парою вершин визначається як округлена до цілого числа евклідова відстань між відповідними цій парі точками площини. AMPL-реалізація задач (1.3.1) – (1.3.10) і (1.3.11) – (1.3.18) для графа st70.tsp (містить 70 вершин) з бібліотеки TSPLIB наведена в додатку Б. У додатку також наведено і протоколи розв'язання обох задач для пошуку найкоротших 10-вершинних циклів в графі st70.tsp за допомогою програм Gurobi 9.1.1 і CPLEX 20.1.0.0 з NEOS-сервера. Розроблену AMPL-реалізацію можна використовувати для знаходження k-вершинних циклів у інших графах з бібліотеки TSPLIB, для чого достатньо дані для графа st70.tsp замінити на дані необхідного tsp-графа.

За допомогою програми Gurobi 9.1.1 з NEOS-сервера можна успішно (за кілька хвилин) розв'язувати задачі (1.3.1) – (1.3.10) і

(1.3.11) – (1.3.18) для 100-вершинних графів. Це підтверджують результати обчислювальних експериментів, наведені в таблиці 1.3.1, для тестових прикладів задачі комівояжера з бібліотеки TSPLIB з кількістю відвідуваних вершин від 70 до 101. Назви задач (графів) наведені в першій колонці таблиці 1.3.1, а в другій колонці вказано кількість вершин відповідного повного графа. У підколонках «N_1», «M_1», «N_2» і «M_2» наведено кількість змінних і кількість обмежень для розв'язуваних задач (1.3.1) – (1.3.10) і (1.3.11) – (1.3.18). З таблиці 1.3.1 видно, що час, витрачений програмою Gurobi 9.1.1 на пошук оптимального маршруту комівояжера, істотно залежить від того, яка форма задачі використовується. Менші з часів виділено жирним шрифтом і вони не перевищують двох хвилин для обраних тестових прикладів.

Табл. 1.3.1. Результати програми Gurobi для розв'язання задач (1.3.1) – (1.3.10) та (1.3.11) – (1.3.18), де $k = n - 1$

Назва графа (TSPLIB)	n	Задача (1.3.1) – (1.3.10)			Задача (1.3.11) – (1.3.18)		
		N_1	M_1	t_{gurobi}	N_2	M_2	t_{gurobi}
st70.tsp	70	9729	5042	**15**	4968	4833	928
eil76.tsp	76	1147	5930	16	5850	5703	**4**
kro100A.tsp	100	1989	10202	**61**	1009	9903	390
kro100E.tsp	100	1989	10202	**116**	1009	9903	870
eil101.tsp	101	2030	10405	38	1030	1010	**27**

Задача знаходження найкоротшого k-вершинного циклу складніша, ніж задача знаходження найкоротшого гамільтонового циклу. Це пояснюється тим, що в ній потрібно визначити підмножину k вершин, для якої буде знайдений гамільтонів підцикл. Це підтверджують наведені в таблиці 1.3.2 результати роботи програм Gurobi 9.1.1 і CPLEX 20.1.0.0 при розв'язанні задачі (1.3.1) – (1.3.10) для п'яти відомих графів kro100A ÷ kro100E з бібліотеки TSPLIB

(тут $n=100$, $k=99$ та $k=49$), назви тестових задач наведені в першій колонці таблиці 3.2. Розрахунки проводилися на NEOS-сервері.

Табл. 1.3.2. Програми Gurobi та CPLEX для розв'язання задачі (1.3.1) – (1.3.10), $s=1$, $k=99$ та $k=49$

Задача	d_{99}^*	t_{gurobi}	t_{cplex}		d_{49}^*	t_{gurobi}	t_{cplex}
kroA100.tsp	21282	48	350		9184	184	178
kroB100.tsp	22141	114	441		9096	241	883
kroC100.tsp	20749	50	901		9307	1622	8343
kroD100.tsp	21294	83	368		8929	396	605
kroE100.tsp	22068	106	506		9312	263	762

З таблиці 1.3.2 видно, що знаходження 49-вершинних циклів вимагає більших витрат за часом, ніж задача комівояжера (99-вершинний цикл). Так, наприклад, для графа kroA100.tsp вони більше в десять разів. Тут витрати часу (в секундах) визначалися для розв'язання задачі (1.3.1) – (1.3.10) і обчислювалися за допомогою функції _solve_time мови AMPL.

1.3.4 Задача про найкоротший k-вершинний шлях

Нехай $D_{n,n}$ – повний граф, де n – кількість вершин, а $d_{ij}>0$ – довжина дуги між вершинами i і j, $i\neq j$. Будемо розглядати орієнтований граф, як на рисунку 1.1.6, що включає граф $D_{n,n}$ і дві додаткові вершини a і b, які не співпадають з вершинами графа $D_{n,n}$. Позначимо $d_{ai}\geq 0$ – довжину дуги, що зв'язує вершину a і вершину i графа $D_{n,n}$, а $d_{ib}\geq 0$ – довжину дуги, що зв'язує вершину i графа з вершиною b.

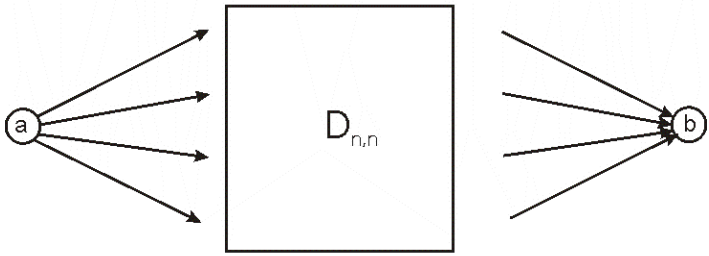


Рис. 1.3.1. Орграф, що включає граф $D_{n,n}$ і вершини a, b

Зв'язний шлях з вершини a до вершини b, який проходить через k вершин графа $D_{n,n}$, де $1 \le k \le n$, будемо називати k-вершинним шляхом з a до b. Якщо $k = n$, то цей шлях проходить через всі вершини графа $D_{n,n}$. Позначимо $(a, i_1, \ldots, i_k, b)$ – послідовність вершин в k-вершинному шляху, де $i_1, \ldots, i_k$ – вершини графа $D_{n,n}$, що йдуть по порядку. Він містить $(k+1)$ дуг, де одна дуга з'єднує вершину a з вершиною i_1, $(k-1)$ дуг послідовно з'єднують вершини $i_1, \ldots, i_k$, і одна дуга з'єднує вершину i_k з вершиною b. Зв'язний k-вершинний шлях, якому відповідає найменша сумарна довжина $(k+1)$ дуг, що входять до нього, будемо називати найкоротшим k-вершинним шляхом, а його довжину позначимо d^*_{abk}.

Розглянемо формулювання задачі змішаного булевого та лінійного програмування для знаходження найкоротшого k-вершинного шляху. Щоб забезпечити зв'язність шляху, тобто уникнути підциклів в графі $D_{n,n}$, використовується ідея моделювання задачі про потік, аналогічно тому, як це зроблено для задачі (1.3.1) – (1.3.10).

Нехай x_{ij} – булева змінна, яка дорівнює одиниці, якщо в шлях входить дуга, яка починається в вершині i і закінчується в вершині j,

і дорівнює нулю в протилежному випадку. Позначимо x_{ai} і x_{ib} – булеві змінні такого ж типу як x_{ij}, але для дуг з вершини a до вершини i і з вершини i до вершини b. Кількість змінних x_{ai}, x_{ib} і x_{ij} дорівнює $n+n+n(n-1)=n(n+1)$. Нехай y_i – булева змінна, яка дорівнює одиниці, якщо шлях проходить через вершину i, та дорівнює нулю в протилежному випадку. Кількість таких змінних дорівнює n. Нехай невід'ємна змінна z_{ij} задає величину потоку певного продукту від вершини i до вершини j, а невід'ємна змінна z_{ai} задає величину потоку від вершини a до вершини i. Кількість цих змінних дорівнює $n+n(n-1)=n^2$.

Знаходженню найкоротшого k-вершинного шляху з вершини a до вершини b відповідає наступна задача:

знайти

$$d^*_{abk} = \min_{y_i, x_{ij}, z_{ij}} \left\{ \sum_{i=1}^{n} d_{ai} x_{ai} + \sum_{i=1}^{n} d_{ib} x_{ib} + \sum_{i=1}^{n} \sum_{j=1,\, j\neq i}^{n} d_{ij} x_{ij} \right\}, \quad (1.3.31)$$

за обмежень

$$\sum_{i=1}^{n} x_{ai} = 1, \quad \sum_{i=1}^{n} x_{ib} = 1, \quad (1.3.32)$$

$$x_{ai} + \sum_{j=1,\, j\neq i}^{n} x_{ji} = y_i, \quad \sum_{j=1,\, j\neq i}^{n} x_{ij} + x_{ib} = y_i, \quad i = 1,\ldots,n, \quad (1.3.33)$$

$$\sum_{i=1}^{n} y_i = k, \quad (1.3.34)$$

$$z_{ai} - k x_{ai} \le 0, \quad i = 1,...,n, \quad (1.3.35)$$

$$\sum_{i=1}^{n} z_{ai} = k, \quad (1.3.36)$$

$$z_{ij} - (k-1) x_{ij} \le 0, \quad i, j = 1,...,n, \quad i \neq j, \quad (1.3.37)$$

$$z_{ai} + \sum_{j=1,\, j\neq i}^{n} z_{ji} - \sum_{j=1,\, j\neq i}^{n} z_{ij} = y_i, \quad i = 1,\ldots,n, \tag{1.3.38}$$

$$x_{ai} = 0 \vee 1,\ x_{ib} = 0 \vee 1,\ i = 1,\ldots,n,\ x_{ij} = 0 \vee 1,\ i, j = 1,...,n,\ i \neq j, \tag{1.3.39}$$

$$y_i = 0 \vee 1, \quad i = 1,\ldots,n, \tag{1.3.40}$$

$$z_{ai} \geq 0, \quad i = 1,...,n, \quad z_{ij} \geq 0, \quad i, j = 1,...,n, \quad i \neq j. \tag{1.3.41}$$

Задача (1.3.31) – (1.3.41) є задачею змішаного булевого лінійного програмування. Вона містить $2n(n+1)$ змінних, з яких $n^2 + 2n$ є булеві, а n^2 – невід'ємні, та $n^2 + 3n + 4$ обмежень, у тому числі $3n + 4$ – лінійні рівності, а n^2 – лінійні нерівності.

Теорема 1.3.3 [17]. *Якщо* k – ціле число, яке задовольняє нерівностям $1 \leq k \leq n$, то обмеження (1.3.30) – (1.3.39) описують всі можливі k-вершинні шляхи з вершини a до вершини b.

Мінімізація цільової функції в (1.3.31) відповідає знаходженню найкоротшого (мінімального за довжиною) шляху з вершини a до вершини b, який проходить через k вершин графа $D_{n,n}$. При цьому d^*_{abk} відповідає довжині найкоротшого шляху. Але сам найкоротший шлях може бути неєдиним, і якщо це так, то розв'язок задачі (1.3.31) – (1.3.41) забезпечує тільки один з можливих найкоротших k-вершинних шляхів з вершини a до вершини b.

1.3.5 Оптимальні k-маршрути для Малопольського винного шляху [18]

Обчислювальний експеримент проводився на задачах відвідування заданої кількості пунктів виноробства з 20 найбільш відвідуваних для напрямку Львів – Вроцлав – Львів (Малопольський винний шлях). Їх розташування наведено на рис. 1.3.2 і характеризується тим, що відстані від Львова і Вроцлава до пунктів виноробства Малопольського винного шляху істотно більші, ніж відстані між пунктами виноробства.

У таблиці 1.3.3 наведені назви пунктів виноробства, d_{ai} – відстані до кожного з них від Львова і Вроцлава, а також d_{ib} – відстані від пунктів виноробства до Вроцлава і Львова. Відстані задані в кілометрах і обчислені за допомогою веб-сервісу Google Maps (https://maps.google.com/).

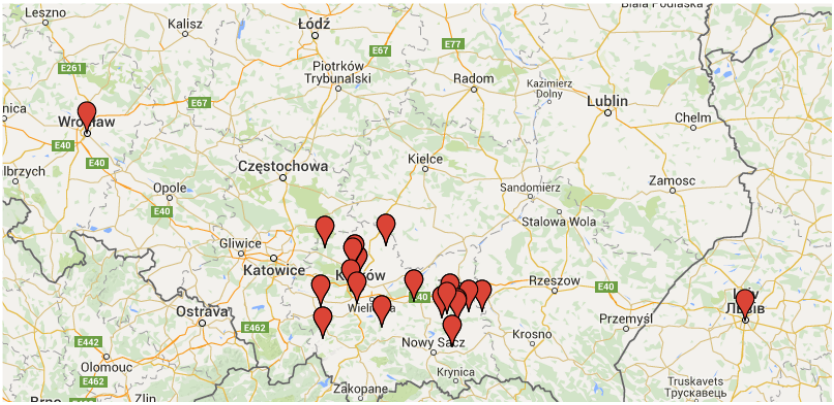


Рис. 1.3.2. Пункти виноробства в напрямку Львів – Вроцлав

Табл. 1.3.3. Відстані Львів – пункти виноробства – Вроцлав

i	Назва пункту виноробства	Львів		Вроцлав	
		d_{ai}	d_{ib}	d_{ai}	d_{ib}
1	Szawapier	386	383	287	286
2	Nad Dobrą Wodą	361	341	267	270
3	Piwnice Antoniego	253	253	389	386
4	Amonit	369	367	274	270
5	Rodziny Stecόw	272	261	369	369
6	Comte	392	393	244	242
7	Kresy	371	369	277	273
8	Krokoszówka Górska	367	367	273	271
9	Hybridium	365	365	264	263
10	Nad Dworskim Potokiem	289	291	319	318

11	Srebrna Góra	351	347	270	263
12	Zadora	276	275	358	354
13	Zawisza	284	285	367	364
14	Kuźnia	247	265	379	375
15	Demeter	266	268	380	377
16	Uroczysko	284	284	366	363
17	Smykań	338	340	315	315
18	Chodorowa	317	319	387	394
19	Dosłońce	341	343	299	295
20	Gaj	340	339	279	275

У таблиці 1.3.4 наведені d_{ij} – відстані між пунктами виноробства (тут індекс $i=\overline{1,n}$ змінюється по горизонталі, а індекс $j=\overline{1,n}$ змінюється по вертикалі). Відстані d_{ij} задані в кілометрах і також обчислені за допомогою веб-сервісу Google Maps.

У таблиці 1.3.5 наведені довжини всіх можливих найкоротших k-вершинних маршрутів для трьох задач відвідування k пунктів виноробства для напрямку Львів – Вроцлав (позначені d^*_{abk}) і для напрямку Вроцлав – Львів (позначені d^*_{bak}). У першій задачі в маршрут могли включатися будь-які з пунктів виноробства, у другій задачі в маршрут обов'язково включався дев'ятий пункт виноробства «Hybridium», а в третій задачі – в маршрут включалися як дев'ятий пункт виноробства, так і дванадцятий пункт виноробства «Zadora».

Розрахунки проводилися за допомогою програми Gurobi на NEOS-сервері (http://www.neos-server.org/neos/solvers/). Знаходження найкоротшого маршруту, що проходить через всі 20 пунктів виноробства вимагало кілька секунд роботи програми Gurobi. Отже, результати обчислювальних експериментів показали можливість використання запропонованої моделі для вибору оптимальних маршрутів в режимі реального часу.

Табл. 1.3.4. Відстані між пунктами виноробства Малопольського винного шляху

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	83.1	172	89.3	152	86.5	93.3	88.8	32.7	101	71.7	140	136	149	149	148	68.4	127	106	46.7
2	80.3	0	145	12.4	108	40.7	15.7	10.2	49.8	51	12.3	96.3	109	105	106	121	55.6	142	44.1	34.3
3	171	126	0	141	12	177	143	151	149	70.9	136	31	40	29	30.7	33	93	56.6	119	125
4	89.4	12.4	145	0	134	32.9	2.2	6.8	58.9	83.1	23.8	122	118	130	132	130	73.6	151	41.8	43.4
5	151	126	12	132	0	157	136	131	129	51.3	116	16	22.8	0.35	13.5	15.8	80	39.4	102	105
6	86.9	41	177	33	157	0	31	39.8	64.1	107	44	145	142	154	156	153	106	174	54.1	67
7	91.4	14.4	156	2	135	31	0	8.8	60.8	85	25.7	124	120	132	134	132	83.9	153	39.9	45.3
8	81	10.2	151	6.8	131	39.8	9	0	56.6	80.8	21.5	120	116	128	130	128	79.7	149	44.9	41.1
9	32.5	51.1	150	59.1	130	63.4	61.3	56.9	0	79.8	41.5	118	115	127	129	127	68.4	148	84.7	31.6
10	100	56.8	65	83	51.7	106	71.4	68	81.9	0	55.1	35.2	35.4	52	49.1	43.2	46.2	68	59.1	54.4
11	71.6	12.2	136	23	116	47	25.9	21.5	41	65.3	0	104	100	113	114	112	64.2	136	54.4	25.6
12	139	114	21	120	16	145	124	119	117	35.1	104	0	19.8	16.4	21.6	10.2	74	45	85.7	93
13	136	110	40	117	22	141	120	116	114	36.2	90.2	19.8	0	23	14.4	14.4	59	33.3	92	89.5
14	148	122	23	129	10.6	154	133	128	126	48.6	103	18.7	13	0	1.3	15.6	70.2	29.6	104	102
15	149	106	29.4	130	12.2	155	134	130	128	50	104	20.3	14.4	1.6	0	12	71.7	28.7	106	103
16	147	122	33.2	130	15.9	153	132	127	125	49.9	112	10.2	14.4	15.6	18.5	0	71.7	44.4	93.9	101
17	68.3	70.3	92	78.4	80	102	80.5	76.1	69.7	46.2	60.7	91.8	58	70.2	71.7	71.7	0	70.6	81.6	36.5
18	127	155	56.6	151	39.4	174	153	148	146	68.9	133	45	33.2	29.6	28.7	47	70.7	0	125	122
19	106	43.9	117	41.9	101	54	40.2	44.9	94	59	56.2	84.7	90.8	103	105	92.7	80.7	123	0	59.7
20	42	35.1	126	41.5	106	66.4	45.3	40.9	31.7	55.1	25.5	93.8	103	102	104	102	36.6	123	78.3	0

Табл. 1.3.5. Довжини найкоротших маршрутів для $k = 1 \div 20$

K	$y_i = 0 \vee 1, i = 1, \ldots, 20$		$y_9 = 1$		$y_9 = y_{12} = 1$	
	d^*_{abk}	d^*_{bak}	d^*_{abk}	d^*_{bak}	d^*_{abk}	d^*_{bak}
1	607	608	628	629	–	–
2	601	615.1	633.9	634.6	656	657
3	613.2	628.2	637	641.7	645.7	654
4	614.3	630.4	639.7	653	645.7	653
5	617	635	643.8	659.9	648.6	659.9
6	628.9	642.1	648	664.6	652.8	664.6
7	639.7	655	659.9	671.5	659.9	671.5
8	643.3	660.3	670.7	681.6	670.7	681.6
9	649	663	683.4	695.4	683.4	695.4
10	658.4	671.9	695	703	695	703
11	670.3	682	716.5	728.3	716.5	728.3
12	681.1	695.8	723	738.4	723	738.4
13	693.8	707	740	752.2	740	752.2
14	705.4	728.2	751.6	764.1	751.6	764.1
15	730.2	743.6	766.3	778.7	766.3	778.7
16	758.5	771.3	779	790.6	779	790.6
17	806	818.3	806.2	818.3	806.2	818.3
18	853.7	865.9	853.7	865.9	853.7	865.9
19	901.9	913.5	901.9	913.5	901.9	913.5
20	964.8	976.8	964.8	976.8	964.8	976.8

1.3.6 Висновки

У розділі розроблено нові математичні моделі задач знаходження оптимальних гамільтонових шляхів і циклів для підграфів заданої потужності в орієнтованому графі. Окремим їх випадком є задача пошуку найкоротшого гамільтонового циклу, що рівносильно відомій задачі комівояжера. Задачі (1.3.1) – (1.3.10) і (1.3.11) – (1.3.18) можуть

бути легко адаптовані для визначення кільцевих маршрутів при плануванні пасажирських і вантажних перевезень. І якщо за допомогою обмежень вигляду (1.3.5) – (1.3.7) легко врахувати транспортування заданих обсягів вантажу, то за допомогою обмежень вигляду (1.3.15) легко врахувати порядок відвідування деяких пунктів призначення.

Задача (1.3.31) – (1.3.41) дозволяє знаходити як найкоротший гамільтонів шлях у повному графі, так і найкоротші гамільтонові шляхи в підграфах, які отримуються вирізанням з повного графа з n вершинами таких повних підграфів, які містять k вершин, де $1 < k \le n$. Для цього досить у формулюванні задачі (1.3.31) – (1.3.41) або покласти всі відстані d_{ai} і d_{ib} рівними нулю, або прибрати внесок у цільову функцію (1.3.31) від тих дуг, які зв'язують вершини a і b з вершинами повного графа.

Розроблені математичні моделі задач знаходження найкоротших k-вершинних шляхів і циклів можуть бути використані при проектуванні і компонуванні технічних об'єктів, оптимізації транспортування товарів, аналізі та прогнозуванні економічних процесів, визначенні оптимальних маршрутів при плануванні пасажирських і вантажних перевезень. Розроблене програмне забезпечення на мові моделювання AMPL може бути використано для розв'язання інших класів прикладних задач оптимізації, наприклад, оптимальної організації процесу управління множиною транзакцій та запитів при їх реалізації у мережевих базах даних [19].

Список літератури

1. Кристофидес Н. Теория графов. Алгоритмический подход. М: Мир. 1978. 432 с.
2. Стецюк П.І., Соломон Д.І., Григорак М.Ю. Задачі про найкоротші k-вершинні цикли та шляхи. Кібернетика та комп'ютерні технології. 2021. № 3. С. 15–33.

3. Gavish B., Graves S.C. The travelling salesman problem and related problems. – Working Paper OR-078-78, 1978. Operations Research Center, MIT, Cambridge, MA.
4. Miller C.E., Tucker A.W., Zemlin R.A. Integer programming formulation of travelling salesman problem. *J. ACM*. 1960. No 3. P. 326–329.
5. Алексеева Е.В. Построение математических моделей целочисленного линейного программирования. Примеры и задачи: Учеб. Пособие. Новосиб. гос. ун-т. Новосибирск. 2012. 131 с.
6. Гамецкий А.Ф., Соломон Д.И. Исследование операций. Том II. Кишинэу, Эврика. 2008. 592 с.
7. Стецюк П.И. Формулировки задач для кратчайшего k-вершинного пути и кратчайшего k-вершинного цикла в полном графе. Кибернетика и системный анализ. 2016. № 1. С. 78–82.
8. Стецюк П.И. Две задачи о кратчайшем k–вершинном цикле // Матеріали XIII Міжнародної науково–практичної конференції «Математичне та програмне забезпечення інтелектуальних систем (MPZIS–2015)» (18–20 листопада 2015 р., м. Дніпропетровськ). С. 215–221.
9. Стецюк П.И., Соломон Д.И. Вычислительные аспекты задачи коммивояжера. Тези доповідей VIII Міжнародної науково-технічної конференції «Інформаційно-компьютерні технології 2016» (22–23 квітня 2016 року). Житомир: ЖДТУ. 2016. С. 91–92.
10. Solomon Dumitru. Transporturi rutiere de mărfuri și pasageri. Cartea I. Organizarea transporturilor rutiere de mărfuri. (Proiect de an) Ch.: Evrica. 2014. 170 p.
11. Стецюк П.І, Супрун А.А. Прискорення gurobi та cplex для задачі комівояжера. Міжнародний сателітний симпозіум «Інтелектуальні рішення–2021», 29 вересня 2021. С. 144–145.
12. Стецюк П.І, Нуриев У.Г., Нуриева Ф.У. Новая модель целочисленного линейного программирования для задачи

коммивояжера. Материалы VII-ой международной научной конференции «Моделирование, методы оптимизации и информационные технологии», Кишинэу, 15-19 ноября 2021 г. С. 315–321.
13. Gurobi Optimization, Inc., Gurobi Optimizer Reference Manual, 2014. https://www.gurobi.com
14. CPLEX Optimizer. High-performance mathematical programming solver for linear programming, mixed-integer programming and quadratic programming. https://www.ibm.com/analytics/cplex-optimizer
15. NEOS Solvers [e-resource]: https://neos-server.org/neos/solvers/
16. Fourer R., Gay D., Kernighan B. AMPL, A Modeling Language for Mathematical Programming. Belmont: Duxburry Press. 2003. 517 p.
17. Стецюк П.И., Лефтеров А.В., Федосеев А.И. Кратчайший k-вершинный путь. Компьютерная математика. Киев: Ин-т кибернетики им. В.М.Глушкова НАН Украины. 2015. № 2. С. 3–11.
18. Стецюк П.И., Лефтеров А.В., Лиховид А.П., Федосеев А.И. Оптимизационный сервис для выбора винных маршрутов. Математическое моделирование, оптимизация и информационные технологии: материалы 5-й Междунар. науч. конф. (Кишинэу, 22–25 марта 2016 г.). Кишинэу: Эврика, 2016. Т. II. С. 337–344.
19. Андрианова Е.Г., Раев В.К., Фильгус Д.И. Определение кратчайших гамильтоновых путей в произвольных графах распределенных баз данных. Российский технологический журнал. 2019. Т. 7. № 4. С. 7–20. https://doi.org/10.32362/2500-316X-2019-7-4-7-20.

1.4 ДВОЕТАПНА ТРАНСПОРТНА ЗАДАЧА З ОБМЕЖЕННЯМ НА КІЛЬКІСТЬ ПРОМІЖНИХ ПУНКТІВ

П. І. Стецюк, О. М. Хом'як, О. О. Жмуд, А. А. Супрун

Анотація. Розглядається модифікація двоетапної транспортної задачі для знаходження найекономічнішого плану перевезення продукції від постачальників до споживачів за умови, що кількість проміжних пунктів є обмеженою, та визначено умови, при яких задача має розв'язок. Ця модифікація є актуальною для пошуку раціонального розташування заданої кількості складів з урахуванням визначеного положення постачальників та отримувачів матеріально-технічних засобів. Досліджено ефективність сучасного програмного забезпечення з NEOS-сервера (відомих програм Gurobi та CPLEX) для розв'язання тестових задач середніх розмірів.

Abstract. A modification of the two-stage transportation problem is considered for finding the most economical plan for product transportation from suppliers to consumers, provided that the number of intermediate points is limited, and the conditions under which the problem has a solution are determined. This modification is relevant for finding a rational arrangement of a given number of warehouses, taking into account the determined position of suppliers and recipients of material and technical means. The effectiveness of modern software from the NEOS server (well-known programs Gurobi and CPLEX) for solving medium-sized test problems was investigated.

1.4.1 Вступ

При формулюванні класичної двоетапної транспортної задачі [1–3] мається на увазі, що вантаж перевозиться від постачальників до споживачів тільки через проміжні пункти. Схема функціонування

перевезень вантажу наведена на рисунку 1.4.1. В якості проміжних пунктів можуть виступати посередницькі фірми та різного роду сховища (склади). У підрозділі описана модифікація двоетапної транспортної задачі для знаходження найекономічнішого плану перевезення однорідної продукції від постачальників до споживачів за умови, що кількість проміжних пунктів є обмеженою зверху [4].

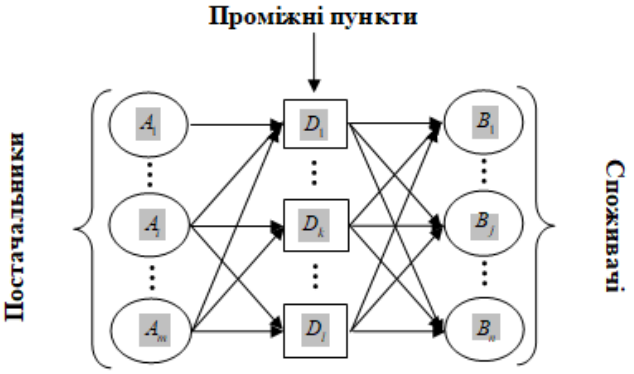


Рис. 1.4.1. Система «постачальники – проміжні пункти – споживачі» у двоетапній транспортній задачі

Вона та її варіанти [5, 6] є актуальними для пошуку раціонального розташування заданої кількості складів з урахуванням визначеного положення постачальників та отримувачів матеріально-технічних засобів. Вона може бути використана також для визначення відповідних місць розташування в ОЕС України накопичувачів електричної енергії та їх енергоємностей.

Модифікація двоетапної транспортної задачі сформульована як задача булевого лінійного програмування. Визначено умови, при яких задача має розв'язок. Досліджено ефективність сучасного програмного забезпечення з NEOS-сервера (відомих програм Gurobi [7] та CPLEX [8]) для розв'язання вказаних тестових задач середніх розмірів (розглядаються декілька сотень постачальників, проміжних пунктів та споживачів).

1.4.2 Формулювання задачі

Нехай в m пунктах постачання $A_1,...,A_m$ є $a_1,...,a_m$ одиниць продукції, яку потрібно перевезти до n споживачів $B_1,...,B_n$, задовольнивши їх потреби $b_1,...,b_n$. Для транспортування продукції від постачальників до споживачів можна задіяти l проміжних пунктів $D_1,...,D_l$, пропускні спроможності яких будемо вважати необмеженими. Потрібно знайти оптимальний план транспортування продукції, який використовує не більше, ніж d $(1 \le d \le l)$ проміжних пунктів, де c_{ik} – витрати на перевезення одиниці продукції від постачальника A_i до проміжного пункту D_k, а c_{kj} – витрати на перевезення одиниці продукції від проміжного пункту D_k до споживача B_j.

Нехай $x = \{x_{ik}\}_{i=\overline{1,m}}^{k=\overline{1,l}}$, де x_{ik} – кількість одиниць продукції, яка перевозиться від постачальника A_i до пункту D_k; $y = \{y_{kj}\}_{k=\overline{1,l}}^{j=\overline{1,n}}$, де y_{kj} – кількість продукції від пункту D_k до споживача B_j; $z = \{z_k\}_{k=\overline{1,l}}$, де z_k – булева змінна, яка дорівнює одиниці, якщо проміжний пункт D_k використовується, та дорівнює нулю – в протилежному випадку.

Двоетапна транспортна задача для знаходження не більше, ніж d проміжних пунктів, які визначають найбільш економічний план перевезення продукції від постачальників до споживачів, має такий вигляд: знайти

$$f_{xyz}^* = f_{xyz}\left(x^*, y^*, z^*\right) = \min_{x,y,z}\left\{ f\left(x,y\right) = \sum_{i=1}^{m}\sum_{k=1}^{l} c_{ik}x_{ik} + \sum_{k=1}^{l}\sum_{j=1}^{n} c_{kj}y_{kj} \right\} \quad (1.4.1)$$

за обмежень

$$\sum_{k=1}^{l} x_{ik} = a_i, \quad i = \overline{1,m}, \tag{1.4.2}$$

$$\sum_{k=1}^{l} y_{kj} = b_j, \quad j = \overline{1,n}, \tag{1.4.3}$$

$$\sum_{i=1}^{m} x_{ik} - \sum_{j=1}^{n} y_{kj} = 0, \quad k = \overline{1,l}, \tag{1.4.4}$$

$$\sum_{i=1}^{m} x_{ik} \le z_k \sum_{i=1}^{m} a_i, \quad k = \overline{1,l}, \tag{1.4.5}$$

$$\sum_{k=1}^{l} z_k \le d, \tag{1.4.6}$$

$$x_{ik} \ge 0,\ y_{kj} \ge 0,\ z_k = 0 \vee 1,\ i = \overline{1,m},\ k = \overline{1,l},\ j = \overline{1,n}. \tag{1.4.7}$$

Задача $(1.4.1) - (1.4.7)$ є задачею булевого лінійного програмування, яка містить $(m+n)\times l$ неперервних змінних x та y, l булевих змінних z, $m+n+2l+1$ обмежень. Цільова функція (1.4.1) задає сумарні витрати на транспортування продукції від постачальників до споживачів. Обмеження (1.4.2) означають транспортування $a_1,...,a_m$ одиниць продукції із пунктів постачання до проміжних пунктів, а обмеження (1.4.3) – що споживачам потрібно доставити необхідні об'єми $b_1,...,b_n$ одиниць продукції з проміжних пунктів. Обмеження (1.4.4) задають умови на те, щоб вся продукція, яка приходить від постачальників до кожного проміжного пункту, була обов'язково відправлена споживачам. Обмеження (1.4.6) означає, що задіяними повинні бути не більше, ніж d проміжних пунктів, а обмеження (1.4.5) визначають, які саме проміжні пункти будуть задіяні.

Якщо для задачі $(1.4.1) - (1.4.7)$ вибрати $d = l$, що означає, що при транспортуванні продукції задіяними можуть бути усі l проміжних пунктів, то тоді обмеження (1.4.5) та (1.4.6) можна прибрати. В результаті отримаємо добре вивчену класичну двоетапну транспортну задачу [2, 3, 9–12]. Тому справедливі наступні

твердження.

Теорема 1.4.1. Якщо $d = l$, то задача (1.4.1) – (1.4.7) переходить в двоетапну транспортну задачу: знайти

$$f_{xy}^{*} = f_{xy}\left(x^{*}, y^{*}\right) = \min_{x,y}\left\{ f\left(x,y\right) = \sum_{i=1}^{m}\sum_{k=1}^{l} c_{ik}x_{ik} + \sum_{k=1}^{l}\sum_{j=1}^{n} c_{kj}y_{kj} \right\} \quad (1.4.8)$$

за обмежень

$$\sum_{k=1}^{l} x_{ik} = a_i, \quad i = \overline{1,m}, \quad (1.4.9)$$

$$\sum_{k=1}^{l} y_{kj} = b_j, \quad j = \overline{1,n}, \quad (1.4.10)$$

$$\sum_{i=1}^{m} x_{ik} - \sum_{j=1}^{n} y_{kj} = 0, \quad k = \overline{1,l}, \quad (1.4.11)$$

$$x_{ik} \geq 0, \ y_{kj} \geq 0, \ i = \overline{1,m}, \ k = \overline{1,l}, \ j = \overline{1,n}. \quad (1.4.12)$$

Якщо в задачі (1.4.8) – (1.4.12) загальна кількість продукції постачальників дорівнює загальному попиту всіх споживачів, тобто виконується рівність

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j, \quad (1.4.13)$$

то таку двоетапну транспортну задачу називають збалансованою, або закритою. Якщо умова (1.4.13) не виконується, то двоетапну транспортну задачу називають незбалансованою або відкритою. Останню легко звести до задачі закритого типу: у разі перевищення загального попиту над запасами – за допомогою введення фіктивного постачальника A_{m+1} із кількістю $a_{m+1} = \sum_{j=1}^{n} b_j - \sum_{i=1}^{m} a_i$ одиниць продукції; у разі перевищення запасів над загальним попитом – за допомогою введення фіктивного споживача B_{n+1} з

потребою $b_{n+1} = \sum_{i=1}^{m} a_i - \sum_{j=1}^{n} b_j$ одиниць продукції.

1.4.3 Про властивості задачі

Згідно з теоремою 1.4.1 задача (1.4.1) – (1.4.7) співпадає з класичною двоетапною транспортною задачею, якщо $d = l$, $1 \le d \le l$. Цей випадок відповідає тому, що d вибирається рівним правій границі інтервалу $[1, l]$ та означає, що при транспортуванні продукції від постачальників до споживачів задіяними можуть бути усі l проміжних пунктів. Якщо $d = 1$, то d вибирається рівним лівій границі інтервалу $[1, l]$, та означає, що всі постачальники відправляють продукцію у єдиний проміжний пункт, з якого кожний споживач забирає свою кількість одиниць продукції. Якщо задача (1.4.1) – (1.4.7) збалансована, тобто виконується умова (1.4.13), то граничному випадку $d = 1$ відповідає єдиний оптимальний план $x^* = \left\{ x_{i1}^* = a_i \right\}_{i=\overline{1,m}}$, $y^* = \left\{ y_{1j}^* = b_j \right\}_{j=\overline{1,n}}$, при якому реалізується мінімальне значення цільової функції $f_{xy} = \sum_{i=1}^{m} c_{i1} a_i + \sum_{j=1}^{n} c_{1j} b_j$.

Якщо виконується умова $1 < d < l$, то тоді задача (1.4.1) – (1.4.7) буде задачею лінійного булевого програмування. В обох крайніх випадках задача булевого лінійного програмування фактично відсутня, так як значення булевих змінних z визначаються однозначно. У першому випадку всі невідомі компоненти z дорівнюють одиниці, а у другому випадку – єдина булева змінна приймає значення, яке дорівнює одиниці.

Теорема 1.4.2. Система обмежень (1.4.2) – (1.4.7) є несумісною,

$$\text{якщо } \sum_{i=1}^{m} a_i \neq \sum_{j=1}^{n} b_j .$$

Теорема 1.4.2 дає можливість перевірити коректність вхідних

даних для задачі (1.4.1) – (1.4.7). Іншими словами, вона дає можливість впевнитися, що задача (1.4.1) – (1.4.7) є збалансованою. Якщо задача є збалансованою, то система (1.4.2) – (1.4.7) є сумісною, і переходимо до розв'язання задачі (1.4.1) – (1.4.7). Якщо задача є незбалансованою, то закінчуємо розрахунок з повідомленням «задача (1.4.1) – (1.4.7) не має розв'язку».

1.4.4 AMPL-реалізація задачі

Один із можливих способів розв'язання задачі (1.4.1) – (1.4.7) полягає у використанні AMPL (A Mathematical Programming Language) [13] та сучасного програмного забезпечення для розв’язання задач цілочислового лінійного програмування, наприклад, програми Gurobi [7].

AMPL-код для опису оптимізаційної задачі (1.4.1) – (1.4.7) має наступний вигляд:

```
param m>=2; # Кількість постачальників (пункти A)
param l>=1; # Кількість проміжних пунктів (пункти D)
param n>=2; # Кількість споживачів (пункти B)
param d>=1<=m; # границя зверху на кількість пунктів D
# Вартості перевезення одиниці продукції:
param cik{i in 1..m, k in 1..l} >= 0; # від A до D
param ckj{k in 1..l, j in 1..n} >= 0; # від D до B
# Інші дані
param a{i in 1..m} >= 0; # Продукція в A
param b{j in 1..n} >= 0; # Потреби в B
# теорема 1.4.1
check: sum{i in 1..m} a[i] = sum{j in 1..n} b[j];
# Невідомі (продукція, яку потрібно перевезти)
var x{i in 1..m, k in 1..l} >=0; # від A до D
var y{k in 1..l, j in 1..n} >=0; # від D до B
var z{k in 1..l} binary;  # 1-задіяно D, 0-незадіяно D
# Мінімізувати витрати на перевезення продукції:
minimize f_opt:
sum{i in 1..m, k in 1..l} cik[i,k]*x[i,k]+ # від A до D
sum{k in 1..l, j in 1..n} ckj[k,j]*y[k,j]; # від D до B
subject to # за обмежень
```

```
con2 {i in 1..m}: # перевезення продукції з A до D
sum{k in 1..l}x[i,k] = a[i];
con3 {j in 1..n}: # задоволення потреб B з D
sum{k in 1..l}y[k,j] = b[j];
con4 {k in 1..l}: # всю продукцію треба доставити в B
sum{i in 1..m}x[i,k]-sum{j in 1..n}y[k,j]=0;
con5 {k in.. 1..l}: # вкл./викл. проміжний пункт
sum{i in 1..m}x[i,k]<=z[k]*sum{i in 1..m}a[i];
con6:  # вибрати не більше ніж d проміжних пунктів
sum{k in 1..l}z[k]<=d;
```

Тут оператор **param** використано для опису розмірів та даних задачі; оператор **var** – для опису змінних задачі, де враховано їх невід'ємність та булевість; оператор **check** – для перевірки сумісності обмежень (1.4.2) – (1.4.7) згідно теореми 1.4.2, яка вимагає, щоб уся продукція постачальників була відправлена споживачам.

Якщо цей AMPL-код доповнити необхідними даними для конкретної задачі, то його можна використовувати для розв'язання задачі (1.4.1) – (1.4.7) за допомогою стандартного програмного забезпечення для розв'язання задач цілочислового лінійного програмування.

1.4.5 Тестовий приклад

Виробниче об'єднання складається з трьох філіалів: A_1, A_2, A_3, які виготовляють однорідну продукцію в обсягах відповідно 200, 220 та 380 одиниць на місяць. Ця продукція відправляється на три склади D_1, D_2 і D_3, а потім – до чотирьох споживачів B_1, B_2, B_3, B_4, попит яких становить відповідно 150, 50, 350 і 250 одиниць. Вартості перевезень одиниці продукції (в умовних одиницях) від виробників на склади, а потім – зі складів до споживачів наведено у наступних таблицях.

$A \setminus D$	D_1	D_2	D_3
A_1	5	2	5
A_2	3	1	3

$D \setminus B$	B_1	B_2	B_3	B_4
D_1	3	2	5	2
D_2	7	1	3	1

A_2	4	7	1

D_3	8	5	6	5

Для тестового прикладу опис вхідних даних задачі (1.1.1) – (1.1.7) реалізує наступний AMPL-код:

```
data; # Блок вхідних даних, де задаємо:
param m := 3; # Кількість постачальників A
param l := 3; # Кількість проміжних пунктів D
param n := 4; # Кількість споживачів B
param d := 3; # границя зверху на кількість D
# Витрати на перевезення одиниці продукції:
param cik: 1 2 3 := # від A ( ) до D ( )
          1 5 2 5
          2 3 1 3
          3 4 7 1;
param ckj: 1 2 3 4 := # від D ( ) до B ( )
          1 3 2 5 2
          2 7 1 3 1
          3 8 5 6 5;
param a:= # Продукція в A
1 200 2 220 3 380; # A_1, A_2, A_3
param b:= # Потреби B
1 150 2 50 3 350 4 250; # B_1, B_2, B_3, B_4
options solver gurobi;
solve; # Розв'язати задачу (1)-(5)
# Роздрукувати значення цільової функції та
# час розв'язання
display f_opt, _solve_time;
# Роздрукувати значення оптимальних змінних
display x; display y; display z;
```

Розв’язок задачі (1.4.1) – (1.4.7) для тестового прикладу при $d = 3$ представлений на рисунку 1.4.2. Для нього значення цільової функції дорівнює 3940 умовних одиниць, склад D_1 завантажений продукцією на 150 одиниць, склад D_2 – на 420 одиниць, а склад D_3 – на 230 одиниць.

Розв’язок задачі (1.4.1) – (1.4.7) для тестового прикладу при $d = 2$

представлений на рисунку 1.4.3. Для нього значення цільової функції дорівнює 4170 умовних одиниць, склад D_1 завантажений продукцією на 380 одиниць, склад D_2 – на 420 одиниць, а склад D_3 – не використовується.

Для тестування сучасного програмного забезпечення для розв'язання двоетапних транспортних задач з обмеженням на кількість проміжних пунктів створено AMPL-коди для відповідних задач булевого лінійного програмування, орієнтовані на велику кількість постачальників, проміжних пунктів та споживачів. Приклад AMPL-коду наведено в [1]. За його допомогою досліджено ефективність сучасного програмного забезпечення з NEOS-сервера (відомих програм Gurobi та CPLEX) для розв'язання тестових задач середніх розмірів (розглядаються декілька сотень постачальників, проміжних пунктів та споживачів).

Наведемо результати розрахунку для тестової задачі з наступними розмірами:

```
param m=100; # Кількість постачальників (пункти A)
param n=250; # Кількість споживачів (пункти B)
param l=100; # Кількість проміжних пунктів (пункти D)
param d >= 1 <= 10; # границя зверху на кількість
# пунктів D
```

Для них за змінними, обмеженнями та заповненістю матриці обмежень відповідна задача булевого лінійного програмування має такі характеристики

```
35100 variables:
    100 binary variables
    35000 linear variables
551 constraints, all linear; 80200 nonzeros
    450 equality constraints
    101 inequality constraints
1 linear objective; 34998 nonzeros.
```

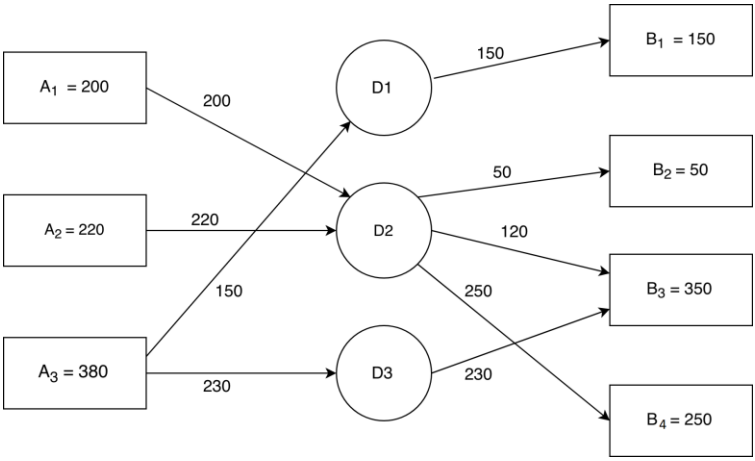


Рис. 1.4.2. Оптимальний план перевезень продукції при $d = 3$

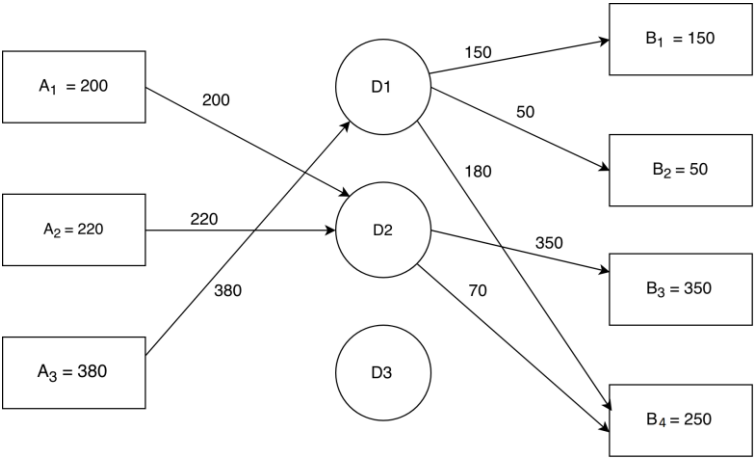


Рис. 1.4.3. Оптимальний план перевезень продукції при $d = 2$

Порівняння часу (в секундах), затраченого програмами Gurobi та CPLEX на розв'язання задач булевого лінійного програмування для різних значень $d \in \{2,...,10\}$, наведено у таблиці 1.4.1. Тут також

наведено максимальну $q_{\max}$ та мінімальну $q_{\min}$ долі завантаження проміжних пунктів при реалізації оптимального плану транспортування продукції.

Табл. 1.4.1. Затрати програм Gurobi та CPLEX для $m = 100$, $l = 100$, $n = 250$

d	f^*_{xyz}	**time gurobi (sec.)**	$q_{\max}$	$q_{\min}$	**time cplex (sec.)**
2	719902	736.731	0.57	0.43	271.351
3	559784	495.499	0.36	0.29	322.735
4	483311	428.549	0.28	0.22	321.833
5	432791	478.035	0.23	0.14	324.89
6	388650	446.238	0.20	0.13	216.514
7	355522	341.772	0.18	0.12	150.851
8	331430	308.822	0.15	0.10	101.462
9	314464	438.228	0.14	0.07	106.149
10	298488	94.3449	0.13	0.06	86.6779

З таблиці 1.4.1 видно, що для булевої двоетапної транспортної задачі (35 000 змінних, 550 обмежень) час розв'язання за допомогою програми CPLEX складає не більше десяти хвилин, що можна використати для розміщення до 10 накопичувачів, які можуть запасати електричну енергію від 100 постачальників та можуть її передавати 250 споживачам. Аналогічні затрати часу для програм Gurobi та CPLEX будуть мати місце, якщо кількість постачальників буде дорівнювати 250, а кількість споживачів − 100.

1.4.6 Висновки

У даному підрозділі наведено результати дослідження двоетапної транспортної задачі для визначення найекономічнішого плану перевезення однорідної продукції від постачальників до споживачів,

коли кількість проміжних пунктів (сховищ, складів) є фіксованою. Якщо в задачі задіяні всі проміжні пункти, то вона збігається з класичною двоетапною транспортною задачею. Наведено математичну модель, сформульовану як задачу булевого лінійного програмування. Визначено умови, при яких задача має розв'язок. Розроблено AMPL-код для її розв'язання сучасними солверами лінійного цілочислового програмування, наведено демонстраційні приклади тестових задач, наведено порівняння результатів роботи відомих програм Gurobi та CPLEX для розв'язання тестових задач середніх розмірів (розглядаються декілька сотень постачальників, проміжних пунктів та споживачів). Відмітимо, що описаний метод не зміниться, якщо обмеження (1.4.7) замінити на обмеження, які враховують нижні та верхні межі на кількість продукції, що транспортується від постачальників до споживачів через проміжні пункти.

Список літератури

1. Розроблення оптимізаційних процедур для задач розташування накопичувачів електроенергії в ОЕС України в сучасних умовах технологічних змін. Етап 1. Розроблення математичних моделей, методів та програмного забезпечення для спеціальних класів двоетапних транспортних задач/ П.І. Стецюк, В.М. Горбачук, О.М. Хом'як та інші. Заключний звіт про науково-дослідну роботу № держ. реєстрації 0119U001641. К.: Ін-т кібернетики імені В.М. Глушкова НАН України. 2019. 81 с.
2. Карагодова О.О., Кігель В.Р., Рожок В.Д. Дослідження операцій: Навч. пос. К.: Центр учбової літератури. 2007. 256 с.
3. Наконечний С.І., Савіна С.С. Математичне програмування: Навч. посіб. К.: КНЕУ. 2003. 452 с.
4. Стецюк П.І., Бисага О.П., Трегубенко С.С. Двоетапна транспортна задача з обмеженням на кількість проміжних пунктів. *Компьютерная математика*. 2018. № 2. С. 119–128.

5. Стецюк П.І., Трегубенко С.С. Про двоетапну транспортну задачу з заданою кількістю проміжних пунктів. Матеріали Двадцятого Міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування» (Кропивницький, 13–14 квітня 2018 р.). За ред. Г.П. Донця. С. 128–132.
6. Стецюк П.І., Трегубенко С.С. Двоетапна транспортна задача з заданою кількістю проміжних пунктів. Тези доповідей IX Міжнародної науково-технічної конференції «Інформаційно-компьютерні технології 2018» (20–21 квітня 2018 р.). Житомир: Вид. О.О. Євенок. 2018. С. 99–100.
7. Gurobi Optimization, Inc., Gurobi Optimizer Reference Manual, 2014. URL: https://www.gurobi.com
8. CPLEX Optimizer.https://www.ibm.com/analytics/cplex-optimizer
9. Стецюк П.І., Міца О.В., Стрелюк О.В., Фесюк О.В. Транспортна задача з обмеженнями на пропускні спроможності проміжних пунктів. Питання прикладної математики і математичного моделювання. Д.: ДНУ, 2017. Вип. 17. С. 207–219. https://pm-mm.dp.ua/index.php/pmmm/article/view/214
10. Стецюк П.І., Мазютинець Г.В., Мілешовський Б.І. AMPL-реалізація двоетапної транспортної задачі. Тези XV Ювілейної Міжнародної науково-практичної конференції “Математичне та програмне забезпечення інтелектуальних систем”, Дніпровський національний університет ім. Олеся Гончара, Дніпро, 22–24 листопада 2017. С. 186–191.
11. Стецюк П.І., Ляшко В.І., Мазютинець Г.В. Двоетапна транспортна задача та її AMPL-реалізація. *Наукові записки НаУКМА. Комп'ютерні науки*. 2018. Т. 1. С. 14–20.
12. Стецюк П.І., Лиховид О.П., Супрун А.А. Про лінійну та квадратичну двоетапні транспортні задачі. *Кібернетика та комп’ютерні технології*. 2020. № 4. С. 5–14.
13. Fourer R., Gay D., Kernighan B. AMPL, A Modeling Language for Mathematical Programming. Belmont: Duxburry Press. 2003. 517 p.

Розділ 2. МАТЕМАТИЧНІ МЕТОДИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ З УРАХУВАННЯМ ESG-ЧИННИКІВ

2.1 МЕТОДОЛОГІЯ БАГАТОФАКТОРНОЇ ОЦІНКИ ІНВЕСТИЦІЙНИХ ПРОЄКТІВ СТАЛОГО РОЗВИТКУ НАЦІОНАЛЬНОЇ ЕКОНОМІКИ

А. А. Гонта, М. Ю. Григорак, О. І. Воловик

Анотація. Узагальнено принципи та ESG-чинники сталого фінансування, запропоновано структурну модель взаємозв'язків ESG-ризиків та кредитних ризиків через призму економічної діяльності суб'єктів сталого інвестування. Сформовано сукупність методів та моделей оцінювання ESG-ризиків та розроблено дерево рішень щодо відбору інвестиційних проектів сталого розвитку з урахуванням таксономії сталої економічної діяльності та пріоритетів регіонального розвитку у повоєнний час з використанням моделей скринінгу та скорингу.

Abstract. The principles and ESG-factors of sustainable financing are summarized, a structural model of interconnections between ESG-risks and credit risks through the lens of sustainable investment entities economic activities is proposed. A set of ESG-risks assessment methods and models is formed, and a decision tree is developed for selection sustainable development investment projects taking into account the taxonomy of sustainable economic activities and post-war regional development priorities, using screening and scoring models.

2.1.1 Вступ

Одним із базових принципів відбудови критичної інфраструктури та економіки України визначено принцип «зеленості», що означає

активне впровадження «зелених» технологій в різних секторах одночасно з метою зменшення викидів вуглецевих сполук та ефективного використання наявних природних ресурсів. З точки зору сталого фінансування при розробці і виборі інвестиційних проектів необхідно враховувати сукупність ESG-чинників та їх складових [1–3], а саме:

- відповідальне ставлення до навколишнього середовища (E – environment). Екологічні принципи визначають те, як підприємства піклуються про навколишнє середовище, як намагаються мінімізувати збитки, які завдають довкіллю;
- висока соціальна відповідальність (S – social). Суть соціальних принципів у тому, як підприємства ставляться до персоналу, своїх постачальників, клієнтів, які умови праці створюються для працівників, у яких соціальних проектах беруть участь тощо;
- висока якість корпоративного управління (G – governance). Управлінські принципи стосуються якості управління підприємствами, а саме прозорість звітності, відносин з акціонерами, зарплати менеджменту тощо.

Кожна складова сукупності ESG-чинників може бути оцінена за допомогою відповідних індикаторів та індексів. Існують системи індикаторів сталого розвитку ООН, Світового банку, ОЄСР, ЄС, Міжнародної знаннєвої платформи зеленого зростання тощо [4, 5]. Ці та інші існуючі методологічні підходи до вимірювання ефективності політики сталого розвитку та рівня досягнення цілей зеленого фінансування потребують значної кількості даних, які часто відсутні у фінансовій чи бухгалтерській звітності. Кінцевий результат роботи оцінки ESG показників залежатиме від того, чи правильно визначені ключові для компанії ESG-фактори, наскільки точно надано їм економічний зміст, який вигляд мають ESG-дані та як вони використовуються під час оцінювання та моніторингу проектів зеленого фінансування.

Використання ESG-показників для реалізації політики сталого фінансування потребує розробки спеціальних методик та кількісних

методів, які би враховували реальність заходів сталого розвитку об’єктів зеленого інвестування та їх схильність до різноманітних ризиків. Фінансові установи, відбираючи проєкти для фінансування, зобов’язані оцінювати їх вплив на довкілля, сталість економічної діяльності та енергоефективність [6, 7].

2.1.2 Взаємозв’язок та взаємовпливи кредитних та ESG-ризиків

В наукових працях [8–10] було досліджено взаємозв’язок фінансових результатів компанії з політикою сталості та ESG-цілями, позитивні ефекти досягнення яких проявляються у зростанні рівня репутації, якості корпоративного управління, клієнтської бази, вартості інтелектуального капіталу та підвищення рентабельності активів в цілому. За даними Комітету Міжнародних стандартів оцінки ESG-фактори становлять свого роду нематеріальний актив підприємства. Разом з тим, негативні ефекти можуть бути пов’язані з додатковими витратами грошових коштів, що зменшує рівень прибутковості бізнесу [11]. Отже, не існує однозначної кореляції між рівнем сталості компанії та її фінансовими показниками інвестиційного проекту. Значимість ESG-чинників має суттєве значення для тих секторів економіки, які характеризуються високою чутливістю до екологічних вимог [12].

Автори [13–15] стверджують, що фірми з високим рівнем ESG характеризуються меншим фінансовим/кредитним ризиком. У цій парадигмі наявність пов'язаної з ESG ризик-премією обумовлюється тим, що фірми з низьким ESG-рівнем характеризуються великими ризиками своєї діяльності. В цілому, емпіричні дослідження дійсно підтверджують наявність ESG-премії і вплив ESG діяльності фірм на дохідність акцій, що розміщуються ними. Особливо виділяється робота [16], в якій автор аналізує не окремі компанії, а цілі фонди і на їх прикладі показує існування ESG-фактора, який пов'язаний виключно з ESG-ризиком.

На рис. 2.1.1 узагальнено взаємозв'язок традиційних кредитних та екологічних ризиків, а також їх вплив на економічну діяльність суб'єктів сталого фінансування. Вихідною передумовою даної структурної моделі є те, що екологічні виклики – це джерело двох груп кліматичних ризиків: ризиків переходу (transition risks) до моделі низьковуглецевої економіки та фізичних ризиків (physical risks). За оцінками NGFS, обидві групи ризиків можуть впливати на економіку/фінанси на мікро- та макрорівнях через ряд каналів трансформації, призводити до значних фінансових втрат [17]. Ризики переходу – можливі фінансові втрати внаслідок швидкого переходу до моделі низьковуглецевої економіки, стрімких трансформацій, зумовлених змінами економічної та екологічної політики, уподобань споживачів, соціальних норм, іміджу, ринкових переваг, розробкою чистих («зелених») технологій. Саме тому вони впливають на прибутковість бізнесу та добробут домогосподарств, породжують фінансові ризики кредиторів та інвесторів. На макроекономіку вони впливають через інвестиції, продуктивність, ціни, особливо, коли перехід призводить до заволодіння активами.

Фізичні ризики – це можливі економічні та фінансові втрати, пов'язані з наслідками змін кліматичного циклу (підвищення середніх температур, зміна рівня води у світовому океані) та екстремальними погодними явищами, зумовленими зміною клімату (сильніші та частіші шторми, повені, посухи тощо). Вони впливають на економіку/фінанси двояко, зокрема, можуть виникнути критичні та хронічні фактори унаслідок екстремальних погодних явищ, можливі дестабілізація бізнесу та пошкодження майна. Такі зміни потребують відповідної адаптації та значних інвестицій від компаній, домогосподарств та урядів.

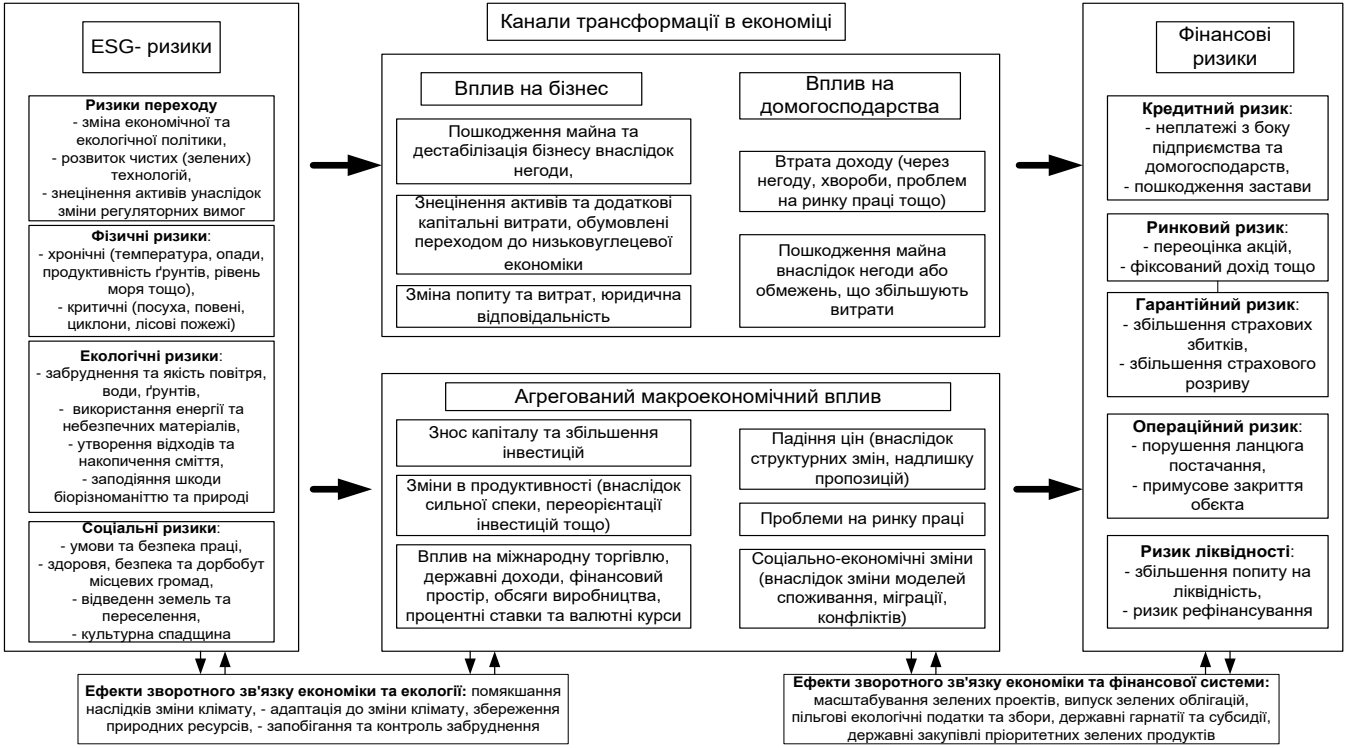


Рис. 2.1.1. Структурна модель взаємозв'язків ESG-ризиків та кредитних ризиків через призму економічної діяльності суб'єктів сталого інвестування (узагальнено на основі [5, 10, 11, 13])

На сьогодні поки що немає єдиної загальновизнаної моделі управління кліматичними ризиками (ризиками переходу і фізичними ризиками) та оцінки їх впливу на економіку. Експерти NGFS вважають, що такі оцінки мають базуватися на публічно доступних даних, які генеруються відібраним набором моделей та відповідним чином узгоджуються. Екологічні та соціальні ризики можуть бути оцінені на основі ESG-чинників та ESG-даних.

2.1.3 Постановка задачі управління ESG-ризиками в умовах цифровизації бізнесу

Проактивне управління показниками ESG-даними допомагає управляти ризиками та створює можливості, а ESG-оцінка є важливою у процесі прийняття рішень, адже допомагає визначити профіль ризику та, як результат, його загальний рівень. Ефективним інструментом збору, аналізу та оцінювання якості ESG-даних є цифрові платформи.

Ідея цифрових платформ для розширення рамок «зеленого» фінансування за рахунок використання нових фінансових технологій («fintech»), зокрема цифрових фінансів (digital finance), у т.ч. для мобілізації коштів дрібних і середніх інвесторів і надання позик невеликим компаніям, стає все більш популярною у світі [18, 19]. З метою досягнення даної мети на щорічному МЕФ у Давосі у 2017 р. був створений Альянс сталого цифрового фінансування (Sustainable Digital Finance Alliance), який працює над проектами залучення коштів через Інтернет для фінансування низьковуглецевих виробництв.

Основним призначенням цифрових платформ сталого фінансування є сполучення підприємств, що сповідують принципи сталого розвитку, з потенційними інвесторами. Вони можуть пов’язувати інституційні інвестиції з добре структурованими проектами та сприяти державному та приватному фінансуванню сталої інфраструктури. Ці типи наскрізних цифрових платформ можуть пришвидшити дуже ручні та неефективні процеси документування, потоки угод і співпрацю між партнерами екосистем. І

вони можуть зробити це з додатковою прозорістю. Це підвищує довіру, зменшує дублювання та забезпечує критично важливу безпеку та можливість перевірки, необхідні для покращеного звітування.

Зауважимо, що, як правило, цифрові платформи сталого (зеленого) фінансування використовують новітні інформаційні технології, блокчейн, «Інтернет речей» та штучний інтелект, що дозволяє прискорити інтеграцію фінансової системи та секторів реальної економіки, розширити можливості для переходу до сталого розвитку. Протягом багатьох років за допомогою машинного навчання фінансові установи успішно виявляли шахрайство з кредитними картками, пов'язуючи інформацію про торгівлю з іншою інформацією щодо поведінки, такою як трафік електронної пошти, записи в календарі, час прибуття та виходу з офісу та навіть телефонні дзвінки.

Наразі цифрові платформи на основі штучного інтелекту можуть управляти ризиками, пов’язаними з реалізацією зелених інвестиційних проектів, оскільки узагальнюють різноманітну інформацію про підприємства, починаючи з їхнього географічного та геополітичного середовища, до фінансових ризиків, стійкості й оцінки соціальної відповідальності компанії. Системи штучного інтелекту можна навчити виявляти, відстежувати та відбивати кібератаки. Ще більші можливості відкриваються для фінансових установ у поєднанні алгоритмів штучного інтелекту з віртуальною реальністю, космічними технологіями (pacetech) та зеленими інноваціями (cleantech). На думку експертів [20] ці технології можуть знайти застосування в режимі реального часу під час:

- аналізу фінансового стану та стійкості бенефіціара або фінансового посередника;
- налагодження взаємовідносин між суб’єктами інвестиційного кредитування та реалізації інвестиційного проекту, враховуючи поведінкові аспекти;
- виявлення відхилень або невідповідностей в інвестиційному проекті за багатофакторною динамічною системою оцінки його ефективності з подільним прийняттям заходів їх нівелювання;

- моделювання та оцінки ризиків інвестиційного проекту;
- генерування рекомендацій і прийняття рішень, гнучких за рахунок використання алгоритмів машинного навчання;
- оптимізації бізнес-процесів за фазами життєвого циклу інвестиційного проекту шляхом зниження операційних, екологічних та фінансових витрат,
- оцінювання інвестиційних портфелів як окремих фінансових установ, так і за основними сферами сталого фінансування,
- краще зрозуміти інвестиційні розриви та потреби у сталому фінансуванні.

Використання технології блокчейну забезпечує здатність цифрової платформи відслідковувати та встановлювати механізми ведення єдиних розподілених реєстрів ліцензування та реєстрація суб'єктів господарювання й інвесторів; випуску цінних паперів (акцій, зелених облігацій); інвестиційних проектів у динаміці результатів їх реалізації, що значно скорочує операційні витрати, час розроблення проектної документації та процедур погодження і затвердження, мінімізує ризики.

Одним із ключових завдань цифрової платформи щодо відбору проектів для сталого фінансування є визначення та стандартизація підходів і методологічних принципів, що дозволяють оцінювати ефективність проектів сталого фінансування, а також проактивне управління ризиками, що вимагає виявлення, вимірювання, моніторингу, контролювання, звітування та пом'якшення екологічних та соціальні ризиків, спричинених підготовкою й реалізацією інвестиційного проекту та діяльністю суб'єкта фінансування. Оскільки заявок на отримання сталого фінансування може бути безліч, то необхідно розробити алгоритм відбору потенційних об'єктів інвестування для подальшого детального дослідження. У цьому випадку доцільно використати модель «воронки» С. Уілрайта та К. Кларка [21], що передбачає використання фільтрів для відсіювання окремих заявок за певними параметрами на основі інструментів скринінгу та скорингу.

2.1.4 Розробка процедури скринінгу та скорингу зелених інвестиційних проєктів

Для кращого розуміння процедури відбору інвестиційних проектів з використанням методів скринінгу та скорингу запропоновано наступну процедуру оцінювання та відбору інвестиційних проектів з точки зору їх відповідності цілям сталого розвитку (рис. 2.1.2).

Першочергово встановлюється відповідність певного виду діяльності технічним критеріям істотного внеску, які можуть включати кількісні або якісні порогові значення. Такий підхід гарантує, що економічна діяльність або має значний позитивний вплив на довкілля, або суттєво зменшує негативний вплив на довкілля. Якщо економічна діяльність, пов'язана з інвестиційним проектом, узгоджується з кліматичною нейтральністю та сприяє обмеженню підвищення температури, тоді її можна кваліфікувати як узгоджену з Таксономією ЄС з метою пом'якшення наслідків зміни клімату.

Галузевий фактор відіграє ключову роль при відборі об'єктів сталого фінансування. Згідно рекомендованої таксономії ЄС, будь-яка сфера господарської діяльності має мати власні критерії перевірки відповідності цілям сталого розвитку. Класифікація видів економічної діяльності, яка базується на кодах NACE, передбачає виокремлення галузей, які використовують природні ресурси як актив, необхідний для здійснення діяльності (наприклад, сільське господарство, добувна галузь, лісова чи рибна промисловість), або ті, які використовують технічні засоби, що мають значний вплив на довкілля (наприклад, машинобудування, енергетика, транспорт тощо).

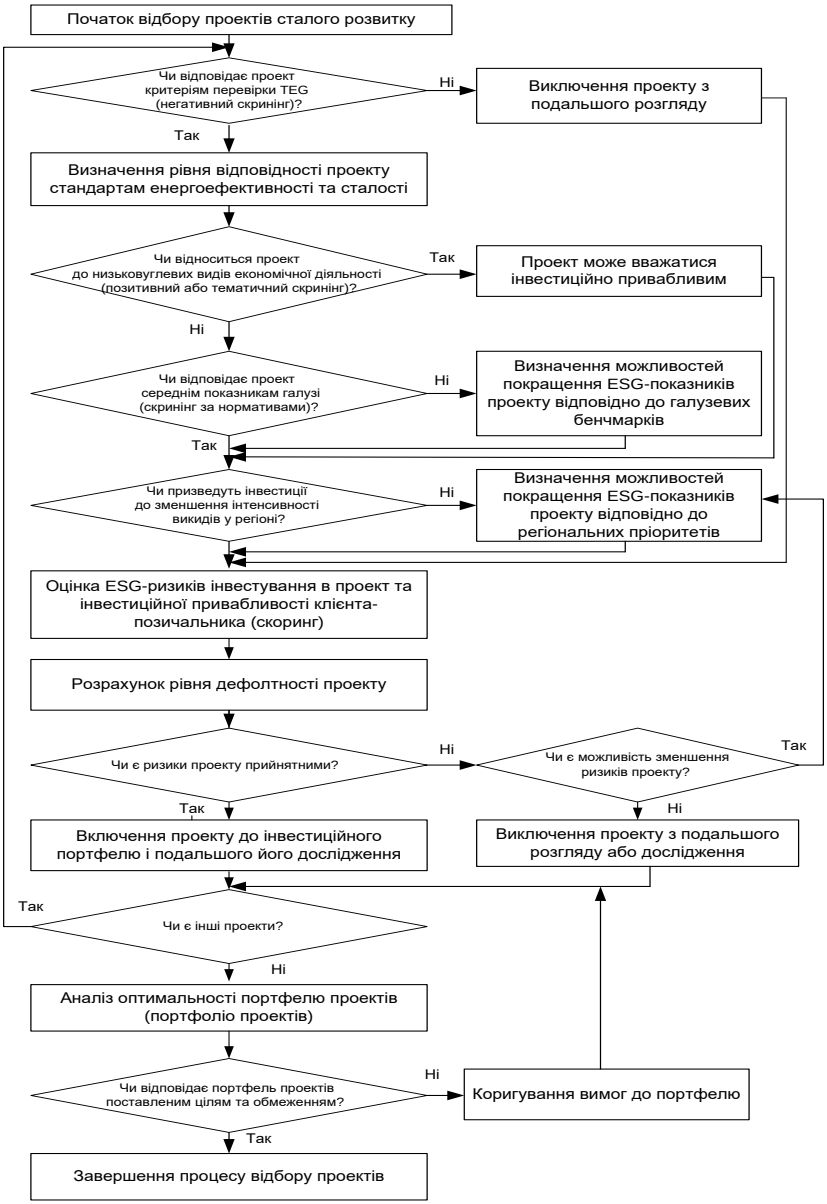


Рис. 2.1.2. Процедура скринінгу та скорингу зелених інвестиційних проектів (авторська розробка)

Існують також галузі, які мають незначний вплив на забруднення довкілля (наприклад, освіта, наука), але вони можуть мати опосередкований вплив за рахунок просвіти та інновацій.

Слід зазначити, що експертами технічної робочої групи TWG визначено пріоритетні галузі з точки зору впливу на довкілля для сталого фінансування [22], а саме:

- сільське господарство, лісове господарство та рибальство;
- гірничо-збагачувальна промисловість;
- виробництво товарів і послуг;
- енергетика,
- будівництво та будівлі;
- транспорт;
- рекреація та туризм;
- водопостачання, водовідведення та управління відходами.

Для цих галузей першочергово мають бути розроблені критерії сталого розвитку та узагальнено кращий досвід технологічних змін зменшення шкідливого впливу на довкілля. Міжнародний досвід свідчить, що найбільші обсяги коштів від емісії “зелених” облігацій спрямовуються у проекти відновлюваної енергетики. Привабливими також залишаються сектор будівництва, сфери енергоефективності та низькоемісійного транспорту. Решту складають сектори водних ресурсів, переробки та утилізації відходів, сільське, лісове та водне господарство.

Наступний етап вимагає аналізу впливу проекту з точки зору принципу «Не завдай значної шкоди», оскільки економічна діяльність може переслідувати екологічну мету, але водночас вона може бути шкідливою за своїми наслідками для інших цілей сталого розвитку або екологічних цілей. Тому такий аналіз також розглядається як подвійне оцінювання певної економічної діяльності з огляду на її сталий характер. Ці аналізи виконуються з використанням процесу комплексної перевірки, коли дана економічна діяльність має відповідати технічним критеріям оцінювання відповідності принципу «Не завдай значної шкоди» Таксономії ЄС. Щоб продемонструвати це,

компаніям та інвесторам необхідно розкрити інформацію про головні несприятливі впливи і встановити прийнятні допуски щодо конкретних показників головних несприятливих впливів, встановлених нормативними технічними стандартами (RTS) Регламенту SFDR. Показники головних несприятливих впливів – це набір попередньо визначених обов'язкових і додаткових точок даних, таких як вуглецевий відбиток, показники води, відходи, політика щодо прав людини, які допомагають компаніям та інвесторам узгоджувати свої операційні та інвестиційні дані з встановленими технічними показниками.

Акцентуємо увагу на тому, що принцип «Не завдай значної шкоди» визначено у статті 17 Таксономії ЄС. Вважається, що діяльність завдає «значної шкоди», якщо вона:

- призводить до посилення несприятливого впливу на поточний клімат та очікуваний клімат у майбутньому, на саму діяльність або на людей, природу чи активи;
- завдає шкоди належному стану або належному екологічному потенціалу водойм, у тому числі поверхневим та підземним водам, або належному екологічному стану морських вод;
- завдає значної шкоди циркулярній економіці, у тому числі запобіганню утворенню відходів та їхній переробці, якщо вона призводить до значної неефективності використання матеріалів або прямого чи непрямого використання природних ресурсів, або якщо вона значно збільшує утворення, спалювання або утилізацію відходів, або якщо тривала утилізація відходів може завдати значної та тривалої шкоди довкіллю;
- завдає значної шкоди запобіганню забрудненню та контролю за ним, якщо вона призводить до значного збільшення викидів забруднювальних речовин у повітря, воду чи землю;
- завдає значної шкоди належному стану та стійкості екосистем або завдає шкоди статусу збереження середовищ проживання та видів, у тому числі тих, що становлять інтерес для Європейського Союзу.

Якщо проект належить до низьковуглецевих або має нульовий рівень викидів вуглецю, то такий проект можна зразу вважати інвестиційно привабливим і досліджувати на ризикованість. Якщо проект відноситься до тих видів діяльності, що дозволяють суттєво зменшити шкідливий вплив на екологію чи є соціально значимим, то його доцільно дослідити з точки зору критеріїв і стандартів галузі та пріоритетів регіонального розвитку.

На нашу думку, регіональний фактор – надзвичайно важливий з точки зору відбору суб'єктів сталого фінансування, оскільки цільові орієнтири регіональної політики та шляхи їх досягнення зумовлюють не тільки різний стан економічних передумов та нерівномірність регіонального розвитку, а й визначають можливості ефективного використання ресурсів для проведення розширеного відтворення і підвищення рівня добробуту населення. Регіон, виконуючи функцію консолідації зусиль з декарбонізації економіки та ресурсоефективності, зможе підвищити або зменшити екологічну, соціально-екологічну, технічну та економічну ефективність господарської діяльності підприємств. Привабливість регіону для залучення «зелених» інвестицій можна визначити з точки зору стану довкілля (обсягів забруднення повітря, води і ґрунтів), утворення відходів життєдіяльності та їх переробки, поточних витрат на охорону навколишнього середовища та підвищення добробуту населення, корпоративної соціальної відповідальності підприємств та організацій тощо. Звичайно, що пріоритетними територіями повоєнної відбудови України будуть ті регіони, які найбільше постраждали від військових дій та потребують нагального відновлення критичної інфраструктури.

Наступним кроком відбору інвестиційних проектів є їх кількісна та якісна ESG-оцінка. Кількісний підхід заснований на публічній інформації про діяльність компанії відповідно до міжнародних стандартів екологічної звітності. Якісний підхід базується на даних, отриманих за допомогою опитувальників, включаючи зовнішні джерела. Дотримуючись першого шляху та вимірюючи ефективність ESG виключно за допомогою інформації, декларованої самою компанією, основний ризик полягає в тому, щоб отримати неповну

оцінку через відсутність даних або часткового звітування. Навпаки, враховуючи інтеграцію альтернативних даних, отриманих зовнішніми зацікавленими сторонами компанії, можна отримати справді неупереджену та правдиву точку зору на параметри сталості інвестиційного проекту. Обидва підходи передбачають використання різноманітних математичних методів, що будуть розглянуті в наступному параграфі.

Процес належної перевірки ESG-діяльності підприємств та окремих інвестиційних проектів повинен відповідати бізнес-моделі банку/фінансової установи та структурі портфеля. Тобто, якщо портфель включає багато різних видів діяльності та значну кількість клієнтів, це означає, що процес екологічної та соціальної оцінки повинен охоплювати більшу частину портфеля, для пом'якшення цих ризиків. В роботі [23] представлено комплексний огляд понад 140 статей щодо формування та оптимізації портфеля проектів. Отримані результати засвідчили, що дослідники останніми роками зосереджувалися не лише на фінансових критеріях, але й на соціальних та екологічних аспектах цілей та завдань компаній, що претендують на отримання інвестицій в зелені технології. При цьому система прийняття рішень щодо сталого фінансування має враховувати той факт, що накопичений фонд заявних коштів, які потребують усі проекти-кандидати, значно перевищує наявні інвестиційні ресурси, тому необхідно аналізувати та оцінювати ESG-ризики.

2.1.5 Аналіз математичного апарату якісного і кількісного оцінювання зелених інвестиційних проєктів

Оцінка ризиків є процесом, який потребує систематичного моніторингу та оновлення. Після розробки стратегій управління ризиками, важливо встановити механізми моніторингу, щоб відстежувати реалізацію цих стратегій та виявляти можливі нові ризики, які можуть виникнути протягом життєвого циклу проєкту.

Рис. 2.1.3. Класифікація моделей для оцінювання проектів сталого фінансування (допрацьовано на основі [24, 25])

Спираючись на досвід банківської системи, зрозуміло, що разом із стрімким розвитком «зеленого» фінансування, необхідно удосконалювати скринінгові та скорингові моделі оцінки проектів. А також оцінювання та моніторингу потенційних ризиків, у тому числі екологічних та соціальних.

На рис. 2.1.3 представлено узагальнену схему використання різних інструментів сталого фінансування, методів якісного та кількісного оцінювання інвестиційних проектів, а також формування портфелю проектів для «зелених» інвестицій з урахуванням ESG-чинників, які можуть створити методологічний базис системи прийняття рішень.

Зауважимо, що скринінг проєктів та інвестиційного портфеля – це вибір із наявної сукупності інвестицій, які відповідають певним критеріям перевірки.

Існує чимало різноманітних критеріїв скринінгу, які дозволяють інвесторам сканувати та виявляти ті компанії, які є фінансово стійкими та/або недооціненими [26–28]. Серед найбільш поширених слід відзначити наступні підходи:

- негативний скринінг – виключення з портфелю певних секторів, компаній або практик за окремими ESG-критеріями, перш за все тих видів діяльності, які не відповідають Таксономії ЄС. Такий вид відбору називають відсіюванням. Він часто ґрунтується на певних стандартах, таких як десять принципів Глобального договору ООН щодо прав людини, трудових стандартів, навколишнього середовища та протидії корупції;
- позитивний скринінг (relative screening, best-in-class screening) – це інвестиції в сектори, компанії чи проекти, відібрані за ESG-критеріями щодо аналогічних компаній галузі. Наприклад, він може включати пошук компаній, які забезпечують права працівників, покращують безпеку праці і клієнтів;

- скринінг за нормативами (norm-based screening) – скринінг інвестицій згідно з мінімальними стандартами ведення бізнесу на основі міжнародних норм;
- розподіл за темами – інвестування в активи, що безпосередньо належать до сталих (наприклад, чиста енергетика, зелені технології, стале сільське господарство). Цей підхід часто ґрунтується на потребах, що випливають із економічних чи соціальних тенденцій.

Метод скринінгу встановлює ESG-фільтри, які застосовуються до інвестицій і, як наслідок, можуть скорочувати інвестиційне поле інвестора. Така політика, як правило, діє на рівні компанії або інвестиційного фонду та характеризується наявністю переліку заборонених дій, продуктів та/або послуг і правил, що визначають, які країни, галузі та компанії заборонені для інвестицій.

При визначенні суттєвості ESG-факторів інвестори оцінюють низку аспектів, у тому числі:

1. Галузеві та країнові фактори, до яких відносяться нормативно-правові та технологічні зміни, пов'язані з діяльністю компаній у даній галузі або ринками, на яких вони виробляють чи реалізують продукцію.
2. Чинники, які стосуються діяльності компанії на певному ринку, продуктів та/або послуг компанії, що відкривають нові можливості, пов'язані з тенденціями в галузі ESG і здатні компенсувати галузеві ризики компанії, або ефективного управління екологічними та соціальними питаннями діяльності компанії щодо зниження її галузевих ESG-ризиків.
3. Часові чинники, що передбачають аналіз ESG-факторів у часовому вимірі. Це зумовлено тим, що ESG-фактори, як правило, матеріалізуються з плином часу, мають істотний вплив на результативність компанії та дозволяють значно підвищити ефективність інвестицій у довгостроковій перспективі.

Зазвичай метод скринінгу проводиться до інвестиційного аналізу. Це суперечить методу ESG-інтеграції, який передбачає, що аналіз

фінансової та ESG-інформації є невід'ємною частиною процесу відбору цінних паперів та формування портфеля, і не передбачає жодних обмежень на інвестиції у будь-які компанії, галузі та (або) країни.

Для поглибленого аналізу застосовують скорингові моделі. Ідея кредитного скорингу була запропонована Д. Дюраном у 40-х роках XX ст., яка полягала у класифікації позичальників на групи за різними ознаками. Скоринг (score – бал) – це математична модель у вигляді зваженої суми певних характеристик, за допомогою якої на основі минулого досвіду банк намагається з'ясувати ймовірність того, що конкретний позичальник не поверне вчасно кредит [29]. У ряді робіт скоринг розглядають як математико-статистичну модель, завдяки якій на підставі кредитної історії, анкети-заявки позичальника та інших даних банк визначає ймовірність повернення кредиту позичальником, тобто його надійність [29–31]. На думку А. Камінського, «успішність скорингової моделі пояснюють деякі ключові фактори: неупередженість оцінки (скоринг відмежовує суб'єктивність оцінок, традиційно пов'язану з кредитними рішеннями); стандартизація кредитних оцінок; можливість автоматизації; контроль (завдяки стандартизації кредитних операцій банкам не важко контролювати та відстежувати ефективність кредитних рішень); зростання дохідності (автоматизація знижує витрати на ручне опрацювання заявок на кредит до мінімуму» [32, с. 77]. Слід зазначити, що чимало наукових досліджень спрямовано на удосконалення систем кредитного скорингу на основі критеріїв сталого розвитку [33]. Зокрема. Автори [34] запропонували модель багатокритеріального кредитного скорингу з урахуванням економічних, екологічних та соціальних факторів, а в роботі [35] розширено поняття портфельного ризик-менеджменту в межах трьохкритеріальної оптимізаційної моделі (мінімізації ризику, максимізації очікуваної дохідності та максимізації ESG-скорингу портфеля).

Скорингові моделі можна класифікувати наступним чином:

- аплікаційний або скоринг заявок (application-скоринг) – оцінка інвестиційної привабливості клієнта-позичальника за наданою інформацією упродовж кредитної трансакції,
- скоринг стягнення (collection-скоринг) – визначення найбільш пріоритетних дій у роботі з позичальниками, стан позикових рахунків яких класифіковано як «незадовільний»,
- скоринг поведінки (behavioral-скоринг) – оцінка динаміки стану позикового рахунку клієнта протягом кредитного періоду,
- передпродажний скоринг (pre-sale) – виявлення потенційних потреб клієнта на основі кредитних історій позичальників,
- скоринг шахрайства (flaud-скоринг) – оцінка ймовірності шахрайства потенційного позичальника.

Кожен вид скорингу потребує власного набору критеріїв оцінювання кредитних ризиків, надійності та платоспроможності позичальників.

Скринінгові та скорингові моделі потребують використання різноманітних математичних методів, зокрема, статистичних, оптимізаційних, порівняльних та інших. Ці моделі базуються на різних методах обробки багатовимірної статистичної інформації: факторний і дискримінантний аналіз, множинна і логістична регресії, дерева оптимізації, нейронні мережі та нечіткі множини, метод найближчих сусідів, генетичні алгоритми, комбіновані методи тощо.

Для того, щоб здійснити перевірку кредитоспроможності і надійності позичальника, сучасні кредитні установи використовують різні скорингові моделі та спеціальні комп'ютерні програми, які дозволяють скоротити час опрацювання інформації. Завдяки цьому відповідь на заявку приймається максимально швидко, а іноді майже миттєво. Як правило, вказані методи використовуються в спеціалізованих програмних комплексах, найбільш відомими серед яких є компанії FICO (FICO Score 9), SAS Institute (SAS Credit Scoring), KXEN (KXEN: Scoring), EGAR Technology (EGAR Scoring), Scorto (Scorto™ Loan Desicion), Lime Systems (WebBank), R-Style Ukraine (RSDH: Система оценки финансового состояния), Бизнес

Нейро-Системы (CreditAnalyst), Deductor Credit Scorecard Modeler (Base Group Labs) [36].

За результатами скринінгу та/або скорингу можуть бути використані різні типи стратегії сталого фінансування [3, 37].

Перший тип стратегій стосується рішень щодо ідентифікації компаній, в які інвестують. Найбільш тривала стратегія супроводжує негативний скринінг, що на базі моральних, нормотворчих або етичних принципів передбачає вилучення акцій з гіршими ESG-характеристиками з портфеля. Зокрема, це є виключення за географією (наприклад, з певних регіонів) або в певних галузях, або за окремими видами діяльності (наприклад, тютюнова галузь через вплив товарів компанії на здоров'я) через етичні причини або для запобігання репутаційної шкоди. Інший підхід передбачає узгодження розподілу капіталу і бажаних сталих результатів, що призводить до використання методів позитивного скринінгу (інвестування в топ-компанії ESG-класу) або нормотворчого скринінгу (наприклад, використання принципів відповідального інвестування Глобального договору ООН (UN Global Compact Principles). Тематичне інвестування передбачає використання відповідних інвестиційних інструментів, які сприяють розподілу капіталу безпосередньо за секторами, що мають дозволити одержати вигоди ESG-принципи та відповідальне інституційне інвестування у світі за різними ESG-напрямками (наприклад, відновлювана енергетика) та боргових інструментів такі як «зелені» облігації (які фінансують екологічні проєкти) та соціальних облігацій (для соціальних проєктів). Другий тип стратегій передбачає залучення корпоративного менеджменту та застосовується після інвестування.

Як і зазначалося попередньо, негативний скринінг може обмежити вплив інвестора на реалізацію ESG-змін, оскільки без прав на акції компанії, інвестор не може голосувати. Натомість, через залучення (що є також відомим як активна форма власності або управління активами) інвестори можуть скористатися своєю позицією як часткові власники компаній для покращення управління компаніями або розкриття ESG-даних. Залучення передбачає обговорення ESG-питань

із менеджментом (через приватні зустрічі або листи та діалог під час телеконференцій або виїзних презентацій) або формальне висловлення схвалення або неузгодження шляхом голосування на основі прав на акції. Інвесторів можна залучати індивідуально, у співпраці з іншими інвесторами або шляхом аутсорсингового залучення провайдера послуг.

Третьою стратегією (та можливо найбільш комплексною) передбачається інтегрування, що визначає зміни в традиційних інвестиційних процесах, із метою включення ESG-даних та ESG-аналізу до комплексного оцінювання інвестицій. Відповідно до цього підходу, інвестиційні команди застосовують дані про сталість для створення більш комплексного бачення інвестиційних ризиків та можливостей, коли й не має значення ступінь сталості інвестиційного фонду. Такий підхід передбачає обробку ESG-даних на етапах дослідження, оцінювання забезпечення або структури портфелів, або пізніше, або під час моніторингу та управління ризиками.

Створення та активне використання цифрової платформи дозволять проводити оцінку проектів сталого розвитку в системі «регіон-галузь-підприємства» та отримувати рейтинг регіонів за рівнем їх привабливості з точки зору сталих інвестицій, оцінку ризиків та ступеня дефолтності проектів, що подаються для фінансування, а також розробляти рекомендації щодо галузевих коефіцієнтів та індикаторів сталого розвитку з урахуванням специфіки різних видів економічної діяльності.

2.1.6 Висновки

Усвідомлення необхідності швидкого переходу до зеленої економіки набирає обертів в наукових та бізнесових колах. Принципи та чинники сталого фінансування мають стати ключовими при інвестуванні повоєнної відбудови економіки України. На відміну від традиційних підходів до залучення інвестицій, стале фінансування визначає пріоритети у відповідності до цілей сталого розвитку ООН та Паризької кліматичної угоди. Зміна пріоритетів вимагає від

фінансових установ розробки методології та інструментів оцінювання інвестиційних проектів з точки зору ESG-чинників, ESG-даних та ESG-ризиків, що, в свою чергу, потребує відповідного математичного забезпечення.

Проведений аналіз процедур скринінгу та скорингу дозволив зробити висновок про необхідність використання теорії оптимізації складних соціо-еколого-економічних систем і різноманітних математичних методів на базі сучасних цифрових платформ для порівняльного аналізу проектів в інтегрованій системі чинників «регіон – підприємство – галузь», відбору привабливих з точки зору фінансування проектів, що сприятимуть зменшенню шкідливого впливу на довкілля, а також ефективного управління ризиками під час їх реалізації.

Список літератури

1. Галушкіна Т.П., Мусіна Л.А., Потапенко В.Г., Машков О.А., Курикін С.І. Основні засади впровадження моделі «зеленої» економіки в Україні.
   URL: http://www.dea.edu.ua/img/source/Book/1.pdf
2. Політика щодо розвитку сталого фінансування на період до 2025 року.
   URL: https://bank.gov.ua/ua/news/all/natsionalniy-bank-prezentuvav-politiku-schodo-rozvitku-stalogo-finansuvannya-na-period-do-2025-roku
3. Матос П. ESG-принципи та відповідальне інституційне інвестування у світі: Критичний огляд досліджень; пер. з англ. Львів: Видавництво Львівської політехніки, 2020. 88 с.
4. A European Green Deal.
   URL: https://ec.europa.eu/info/strategy/priorities-2019-2024/european-green-deal_en
5. Platform on Sustainable Finance.
   URL: https://finance.ec.europa.eu/sustainable-finance/overview-sustainable-finance/platform-sustainable-finance_en#what

6. Зінченко О. А. Світові тренди «зеленого» інвестування. *Економічний простір*. 2022. № 177. С. 31–34.
7. Popescu I.-S., Hitaj C., Benetto E. Measuring the sustainability of investment funds: A critical review of methods and frameworks in sustainable finance. Journal of Cleaner Production. 2021. Vol. 314. doi.org/10.1016/j.jclepro.2021.128016.
8. Fama E.F., French K.R. A five-factor asset pricing model. Journal of Financial Economics. 2015. Vol. 116. No 1. Pp. 1–22.
9. Aydogmus M , Gulay G., Ergun K. Impact of ESG performance on firm value and profitability. Borsa Istanbul Review. 2022. Vol.22–S2. P. S119–S127.
10. Duque-GrisalesE., Aguilera-Caracuel J. Environmental, social and governance (ESG) scores and financial performance of multilatinas: Moderating effects of geographic international diversification and financial slack. Journal of Business Ethics. 2021. No 168. P. 315–334.
11. Pham T. N., Tran P. P., Le M. H., Vo H. N., Pham C. D., Nguyen H. D. The effects of ESG combined score on business performance of enterprises in the transportation industry. Sustainability. 2022. No 14(14). P. 8354.
12. Iazzolino G., Bruni M.E., Veltri S., Morea D., Baldissarro G. The impact of ESG factors on financial efficiency: An empirical analysis for the selection of sustainable firm portfolios. Corporate Social Responsibility and Environmental Management. 2023. №2. P. 1–11.
13. Sassen R., Hinze A., Hardek I. Impact of ESG Factors on Firm Risk in Europe. Journal of Business Economics. 2016. Vol. 86. No 8. Pp. 867–904.
14. Hu V., Scholtens B. Corporate Social Responsibility Policies of Commercial Banks in Developing Countries. Sustainable Development. 2014. Vol. 22. No 4. Pp. 276–288.
15. Maiti M. Is ESG the succeeding risk factor? Journal of Sustainable Finance & Investment. 2020. P. 1–15.

16. Jin I. Is ESG a Systematic Risk Factor for US Equity Mutual Funds? Journal of Sustainable Finance & Investment. 2018. Vol. 8. No 1. P. 72–93.
17. NGFS Climate Scenarios for central banks and supervisors. URL: https://www.ngfs.net/en/liste-chronologique/ngfs-publications.
18. Actionable ESG data and benchmarks for financial markets. URL: https://www.gresb.com/nl-en/
19. Recommendations of the Task Force on Climate-related Financial Disclosures. Final Report. TSFD. URL: https://www.fsb-tcfd.org/wp-content/uploads/2017/06/FINAL-2017-TCFD-Report-11052018.pdf.
20. Пантєлєєва Н.М., Пантєлєєва К.О. Цифрова екосистема інвестиційного кредитування. Причорноморські економічні студії. 2019. Вип. 43. С. 151–155.
21. Steven C., Wheelwright S.C., Clark K.B. Revolutionizing product development: quantum leaps in speed, efficiency and quality. Free Press. 2011. 392 p.
22. FAQ: What is the EU Taxonomy and how will it work in practice? URL: https://finance.ec.europa.eu/system/files/2021-04/sustainable-finance-taxonomy-faq_en.pdf
23. Mohagheghi V., Mousavi S. M., Antuchevičienė J., Mojtahedi M. Project portfolio selection problems: A review of models, uncertainty approaches, solution techniques, and case studies. Technological and Economic Development of Economy. 2019. Vol 25(6). P. 1380–1412.
24. «Зелені» інвестиції у сталому розвитку: світовий досвід та український контекст / кер. проекту К. Маркевич; наук. конс. В. Сіденко. Київ: Заповіт, 2019. 316 с.
25. Routledge Handbook of Social and Sustainable Finance. Edited by Othmar M. Lehner. New York. Routledge. 2017. 652 p.
26. Jin I. ESG-screening and factor-risk-adjusted performance: the concentration level of screening does matter. Journal of Sustainable Finance & Investment. 2020. DOI:10.1080/20430795.2020.1837501

27. Grim D.M., Berkowitz D.B. ESG, SRI, and Impact Investing: A Primer for Decision-Making. The Journal of Impact and ESG Investing Fall. 2020. No 1(1). P. 47–65.
28. Eccles R., Rajgopal S., Xie J. Does ESG Negative Screening Work? URL: http://dx.doi.org/10.2139/ssrn.4150524
29. Бучко І.Є. Скоринг як метод зниження кредитного ризику банку. Вісник Університету банківської справи Національного банку України. 2013. № 2(17). С. 178–182.
30. Волик Н. Г. Скоринг як експертний метод оцінювання кредитного ризику комерційного банку при споживчому кредитуванні. Вісник Запорізького університету. Серія «Економічні науки». 2008. № 1. С. 40–44.
31. Мінін А.Л. Основні підходи до створення та впровадження скорингової системи оцінки платоспроможності позичальників. Економіка і управління. 2016. No 3. С. 109–117.
32. Камінський А. С. Експертна модель кредитного скорингу позичальника банку. Банківська справа. 2009. № 1. С. 75–81.
33. Zeidan R., Boechat C., Fleury A. Developing a Sustainability Credit Score System. Journal of Business Ethics. 2015. №127(2). P.283–296.
34. Roy P.K., Shaw K. Developing a multi-criteria sustainable credit score system using fuzzy BWM and fuzzy TOPSIS. Environ Dev Sustain. 2022. №24. P. 5368–5399.
35. Камінський О. Характеристичні особливості інвестиційного ризик-менеджменту в ESG-інвестуванні: дослідження фондових ринків Центральної та Східної Європи. Наукові записки НаУКМА. Економічні науки. 2022. №7(1). P. 54–60.
36. Андренко О. А., Мордовцев С. М., Мордовцев О. С. Інформаційна система кредитного скорингу. БІЗНЕСІНФОРМ. 2019. № 4. С. 341–347.
37. Chițimiea A., Minciu M., Manta A.M., Ciocoiu C.N., Veith C. The Drivers of Green Investment: A Bibliometric and Systematic Review. Sustainability. 2021. Vol 13(6). P. 3507.

2.2 СТАТИСТИЧНІ ТА ОПТИМІЗАЦІЙНІ МЕТОДИ В КРЕДИТНОМУ СКОРИНГУ

В. О. Стовба

Анотація. Зроблено огляд основних математичних методів, які використовуються в задачах кредитного скорингу. В основу огляду покладено матеріал книги «Credit Scoring and Its Applications» 2002 року (L. Thomas, D. Edelman, J. Crook). Розглянуто статистичні та нестатистичні методи, зроблено їх порівняння та описано нюанси їх застосування.

Abstract. A review of the main mathematical methods used in credit scoring tasks is made. The review is based on materials of the book "Credit Scoring and Its Applications" (L. Thomas, D. Edelman, J. Crook). Statistical and non-statistical methods are considered, their comparison is made and the nuances of their application are described.

2.2.1 Вступ

Кредитний скоринг (credit scoring) – один з найдавніших засобів керування фінансовими ризиками. Початок його використання в США та Великій Британії припадає на другу половину XX століття. Кредитний скоринг певною мірою заклав основи глибинного аналізу даних (data mining), оскільки у цій області в одній з перших використовувались дані про поведінку споживачів. Загалом, такі найбільш відомі техніки аналізу даних як сегментація, кластеризація, моделювання схильності, досить успішно використовувались у кредитному скорингу [1].

Кредитний скоринг – одна з областей, де найбільш успішно застосовувались статистичне моделювання та моделювання дослідження операцій у фінансовій галузі та банкінгу [1]. Оскільки скоринг це математична модель, що оперує великими обсягами даних про клієнтів, для її розв’язання протягом останніх років широкого

вжитку набуло використання методів машинного навчання, зокрема, логістичної регресії, методу опорних векторів, методу *k* середніх (алгоритм Ллойда), *k* найближчих сусідів тощо.

У підрозділі зроблено огляд найбільш вживаних математичних методів у області кредитного скорингу, які умовно поділяються на статистичні (дискримінантний аналіз, логістична регресія тощо) та нестатистичні методи (математичне програмування, нейронні мережі, генетичні алгоритми тощо). Наведено порівняння розглянутих методів та описано умови їх застосування до задач кредитного скорингу.

2.2.2 Кредитний скоринг: основні поняття та процеси

Кредитний скоринг – це набір моделей прийняття рішень та їх технік, за допомогою яких вирішується питання щодо схвалення або відмови у наданні позики заявнику. Ці техніки визначають хто отримає позику та якого обсягу, а також які операційні стратегії можуть покращити прибутковість позичальників та позикодавців. Зазвичай позикодавцями приймаються рішення стосовно двох питань – схвалення чи відмови у наданні кредиту новому клієнту, та політики роботи з існуючими клієнтами, зокрема, в питанні підвищення їхніх кредитних лімітів. Вирішення другого питання лежить в області поведінкового скорингу (behavioral scoring) [1].

Для прийняття рішення щодо схвалення чи відмови у наданні кредиту необхідно проаналізувати наявні дані про заявника та його кредитну історію. В переважній більшості технік кредитного скорингу для віднесення певного клієнта до категорії «хороший» (кредитну заявку схвалено) чи «поганий» (кредитну заявку відхилено) використовуються вибірки заявників з певними даними про них. Для оцінки (скорингу) платоспроможності заявника дані необхідно представити у чисельному вигляді. Для цього використовуються скорингові картки (scorecards), які містять набір характеристик заявника. Залежно від відповіді заявника кожна характеристика оцінюється певною кількістю балів. Результат такої скорингової операції це певна числова оцінка платоспроможності заявника, яка й

використовується для прийняття рішення стосовно надання кредиту.

Існує ціла низка методів роботи з цією оцінкою. В одному з найпростіших варіантів встановлюється певний прохідний бал, якщо оцінка вища ніж цей бал, заявник може бути рекомендованим для надання позики. В іншому випадку, якщо оцінка потрапляє у певний встановлений числовий проміжок («сіру зону»), заявник піддається більш ретельному аналізу, можливо, з залученням додаткових даних. Також часто беруть до уваги пріоритет отриманих даних: наприклад, заявник, оцінка якого вища за прохідний бал, але який у минулому був оголошений банкрутом, переходить у групу заявників, запити яких розглядаються окремо з залученням кредитних аналітиків.

Формулювання задачі, що лежить в основі кредитного скорингу, призвела до появи широкого кола методів, які успішно застосовувались для її розв'язання. На початкових етапах розвитку кредитного скорингу єдиними методами, які використовувались, були класифікаційні методи та статистична дискримінація. Пізніше цей набір методів поповнився багатьма іншими статистичними та нестатистичними методами, які виявились найефективнішими. Задачі кредитного скорингу вдалось сформулювати як оптимізаційні задачі, що дозволило застосовувати цілий клас нових методів. Новий підхід у розв'язанні задач класифікації за допомогою нейронних мереж вдалось успішно застосувати для задач кредитного скорингу.

2.2.3 Статистичні методи для побудови кредитних скорингових карток

Статистичні методи володіють переліком важливих властивостей, які дозволяють успішно застосовувати їх до задач кредитного скорингу. Зокрема, ці методи дозволяють оцінювати дискримінантні можливості скорингової картки та відносну важливість різних характеристик, які її утворюють. Це дає можливість вилучити неважливі характеристики та забезпечити оптимальний набір характеристик у картці. Також з'являється можливість з'ясувати які зміни необхідно внести в питання, які задають клієнтам, для побудови

більш якісних оцінок. Розглянемо найбільш вживані в області кредитного скорингу статистичні методи.

Дискримінантний аналіз. Існує три основні підходи до задачі кредитного скорингу, в яких лінійні дискримінантні функції використовуються як класифікатори: теорія прийняття рішень, поділ на дві групи та лінійна регресія. В першому підході здійснюється пошук правила, яке мінімізує очікувані затрати у прийнятті рішення щодо схвалення кредитної заявки. Другий підхід полягає у побудові функції, яка найкращим чином розділяє заявників на «хороших» та «поганих» у контексті надання кредиту. Третій – ґрунтується на побудові рівняння лінійної регресії, мета якого – знаходження найкращої оцінки правдоподібності заявника бути «хорошим».

Теорія прийняття рішень. Нехай $X = \left(X_1, ..., X_p \right)$ – множина з p характеристик заявника на кредит. Кожна характеристика X_i має скінчену кількість можливих значень (атрибутів), з яких заявник обирає одне. Таким чином вектор $x = \left(x_1, ..., x_p \right)$ – фактичний результат опитування, який описує конкретного заявника. Позначимо A множину всіх можливих векторів x, тобто результатів опитування довільного заявника. Задача кредитного скорингу полягає у знаходженні такого правила, яке розділяє множину A на дві підмножини A_G та A_B. Підмножина A_G містить заявників, що класифікуються як «хороші» та отримують кредит, підмножина A_B містить заявників, класифікованих як «погані», при цьому очікувані витрати мінімальні. Для побудови функції сумарних витрат розглядаються помилки класифікації двох типів. В першому випадку «хороший» заявник класифікується як «поганий», при цьому втрачається потенційний прибуток від такого заявника величини L. В другому випадку «поганий» заявник класифікується як «хороший» і внаслідок неспроможності заявника погасити борг втрачається позика величини D. Тоді функція сумарних витрат має такий вигляд:

$$L \sum_{x \in A_B} p(x \mid G) p_G + D \sum_{x \in A_G} p(x \mid B) p_B, \qquad (2.2.1)$$

де p_G та p_B – частки заявників, які є «хорошими» та «поганими» відповідно, $p(x|G)$ та $p(x|B)$ – умовні ймовірності того, що заявник має вектор атрибутів x за умови, що він є «хорошим» або «поганим» відповідно. Правило поділу є таким:

$$A_G = \{x | Dp(x|B) p_B \le Lp(x|G) p_G\} = \left\{x | \frac{D}{L} \le \frac{p(x|G) p_G}{p(x|B) p_B}\right\}, \quad (2.2.2)$$

тобто заявнику з вектором атрибутів x надається кредит, якщо втрати при хибній класифікації його як «хорошого» будуть не більшими, ніж втрати при хибній класифікації заявника як «поганого». Якщо величини L та D є невідомими, мінімізація сумарних витрат замінюється мінімізацією ймовірності допустити помилку одного з типів, зберігаючи ймовірність допустити помилку іншого типу сталою. В такому випадку виникає задача умовної мінімізації, яка може бути розв'язана за допомогою методу множників Лагранжа.

Варто відзначити, що у тому випадку, коли характеристики заявника є не дискретними, а неперервними випадковими величинами, мають місце аналогічні міркування з використанням функцій щільності випадкових величин та операції взяття інтегралу.

Поділ на дві групи. Нехай $Y = \omega_1 X_1 + ... + \omega_p X_p$ – довільна лінійна комбінація характеристик $X = (X_1, ..., X_p)$. Необхідно знайти таку комбінацію характеристик, які найкращим чином розділяють множину заявників на дві групи, що інтерпретується як схвалення або відхилення заявки про кредитування. Роберт Фішер запропонував використовувати таку міру розділення множин:

$$M = \omega^T \frac{m_G - m_B}{\sqrt{\omega^T S \omega}}, \quad (2.2.3)$$

за умови, що вибіркова дисперсія двох груп однакова. Тут m_G та m_G – вибіркові середні груп «хороших» та «поганих» заявників відповідно, S – вибіркова дисперсія цих груп. Використання необхідних та достатніх умов екстремуму дозволяє отримати значення ваг

$$\omega^T \propto S^{-1}\left(m_G - m_B\right)^T, \qquad (2.2.4)$$

що показує незалежність отриманого результату від форми розподілу випадкових величин $X_1,...,X_p$. Геометрично ваги ω^T утворюють гіперплощину $\omega \cdot x = c$, яка розділяє дві стандартизовані групи «хороших» та «поганих» заявників, причому точка прохідного балу знаходиться на перетині цієї гіперплощини та відрізку, що з'єднує середні значення двох груп.

Лінійна регресія. В цьому підході задача кредитного скорингу полягає у знаходженні лінійної комбінації характеристик

$$\omega_0 + \omega_1 X_1 + ... + \omega_p X_p = \omega^* \cdot \left(X^*\right)^T, \qquad (2.2.5)$$

яка найкращим чином описує ймовірність несплати. Тут $\omega^* = \left(\omega_0, \omega_1, ..., \omega_p\right)$ та $X^* = \left(1, X_1, ..., X_p\right)$. Тобто якщо p_i – це ймовірність того, що заявник i не сплатив борг, необхідно знайти вектор ω^* такий, що

$$p_i = \omega_0 + x_{i1}\omega_1 + ... + x_{ip}\omega_p \text{ для всіх } i. \qquad (2.2.6)$$

Для цього мінімізується величина

$$\sum_{i=1}^{n_G}\left(1 - \sum_{j=0}^{p}\omega_j x_{ij}\right) + \sum_{i=n_G+1}^{n_G+n_B}\left(\sum_{j=0}^{p}\omega_j x_{ij}\right)^2, \qquad (2.2.7)$$

де n_G та n_B – кількість «хороших» та «поганих» заявників відповідно, причому $n_G + n_B = n$. Для простоти викладення матеріалу припускається, що для перших n_G заявників у вибірці $p_i = 1$, для решти – $p_i = 0$. Використання необхідних та достатніх умов екстремуму дозволяє отримати розв'язок $S\omega^T = c\left(m_G - m_B\right)^T$.

Загалом, дискримінантний підхід – це один з найбільш розповсюджених та установлених методів розв'язання задачі кредитного скорингу. Одним з найперших застосувань цього підходу був аналіз позик на автомобілі. До більш пізніх робіт можна віднести публікації [2, 3].

Логістична регресія. Одним з недоліків лінійної регресії є те, що права частина регресійних рівнянь набуває значень від $-\infty$ до $+\infty$, а ліва частина інтерпретується як ймовірність, тому має належати відрізку $[0,1]$. Для подолання цього недоліку ліву частину можна представити як функцію від p_i, областю значень якої буде інтервал $(-\infty,+\infty)$. Такий підхід має назву логістична регресія, а відповідне рівняння має такий вигляд:

$$\log\left(\frac{p_i}{1-p_i}\right)=\omega_0+\omega_1 x_1+\ldots+\omega_p x_p=\omega\cdot x^T. \qquad (2.2.8)$$

Нехай μ_G та μ_B – середні значення серед «хороших» та «поганих» заявників відповідно, а Σ – їх коваріаційна матриця. Тоді відповідна функція щільності характеристик X_i, які мають багатовимірний нормальний розподіл, є такою:

$$f(x\,|\,G)=(2\pi)^{-\frac{p}{2}}(\det\Sigma)^{-\frac{1}{2}}\exp\left(\frac{-(x-\mu_G)\Sigma^{-1}(x-\mu_G)^T}{2}\right). \qquad (2.2.9)$$

Якщо p_G та p_B – частки «хороших» та «поганих» заявників відповідно, рівняння логістичної регресії має такий вигляд:

$$\log\left(\frac{p_i}{1-p_i}\right)=\log\left(\frac{p_G f(x\,|\,G)}{p_B f(x\,|\,B)}\right)=x\cdot\Sigma^{-1}2(\mu_B-\mu_G)^T+$$

$$+\left(\mu_G\cdot\Sigma^{-1}\mu_G^T+\mu_B\cdot\Sigma^{-1}\mu_B^T\right)+\log\left(\frac{p_G}{p_B}\right). \qquad (2.2.10)$$

Підхід для визначення коефіцієнтів за допомогою мінімізації суми квадратів відхилень не застосовується до логістичної регресії, натомість можна використати максимізацію функції правдоподібності.

Підхід логістичної регресії – статистично кращий інструмент порівняно з лінійною регресією для розв'язання задачі бінарної класифікації. Це призвело до інтенсивного застосування цього підходу в області кредитного скорингу [4–6].

Класифікаційні дерева. Якісно іншим підходом до розв'язання задач кредитного скорингу є класифікаційні дерева або алгоритми

рекурсивного розділення. Мета таких алгоритмів – це розділення множини заявників на певну кількість підмножин з подальшою їх класифікацією як «хороших» або «поганих» підмножин. Зазвичай класифікація здійснюється так: якщо переважна кількість заявників у підмножині є «хорошою», її класифікують як «хорошу», інакше як «погану». Ще один спосіб класифікації це мінімізація втрат хибної класифікації.

Для роботи таких алгоритмів необхідно визначити три правила.

1. Правило розділення визначає спосіб розділення множини на дві підмножини.
2. Правило зупинки встановлює умову зупинки процесу розділення множин.
3. Потрібне для класифікації термінальних множин на «хороші» та «погані».

Розділення множин здійснюється так: відбувається перебір можливих варіантів розділення з подальшою перевіркою якості кожного варіанту. Для цього вводиться певна міра якості такого розділення. Наприклад, для неперервної характеристики X_i множина розділяється на підмножини $\{x_i < s\}$ та $\{x_i \geq s\}$ для всіх значень s, після чого визначається для якого значення s міра найкраща. Якщо характеристика X_i категорійна змінна, відбувається перебір всіх варіантів розділення множини з перевіркою значень міри на кожному розбитті. Найбільш вживана міра це статистика Колмогорова – Смірнова, однак використовуються також базовий індекс змішаності та індекс Джині, індекс ентропії та максимізація напівсуми квадратів.

Статистика Колмогорова – Смірнова визначається для неперервних характеристик X_i з кумулятивними функціями розподілу $F(s|G)$ та $F(s|B)$ для «хороших» та «поганих» заявників відповідно, і формулюється як задача: знайти величину s таку, що мінімізує

$$LF(s|G)\,p_G + D\big(1-F(s|B)\big)\,p_B\,. \qquad (2.2.11)$$

Тут L та D – втрати за хибну класифікацію першого та другого

типів відповідно. Якщо $Lp_G = Dp_B$, то задача співпадає з задачею знаходження відстані Колмогорова – Смірнова між двома розподілами, тобто мінімізувати $F(s|G) - F(s|B)$ або максимізувати $F(s|B) - F(s|G)$. Якщо дві підмножини, що утворюються у результаті розділення, розглянути як ліву l та праву r множини, задачу можна сформулювати як максимізацію різниці між $p(l|B)$ та $p(l|G)$ – ймовірністю «поганого» заявника потрапити в ліву групу та ймовірністю «хорошого» заявника потрапити у праву групу. Тоді статистику Колмогорова – Смірнова можна використовувати і для дискретних, і для неперервних характеристик X_i у такому вигляді: знайти таке розділення на ліву та праву групи, яке максимізує

$$KS = \left| p(l|B) - p(l|G) \right| = \left| \frac{p(B|l)}{p(B)} - \frac{p(G|l)}{p(G)} \right| \cdot p(l). \quad (2.2.12)$$

Базовий індекс змішаності та індекс Джині. Існує ціла низка індексів, які дозволяють визначити рівень змішаності кожного вузла дерева. Вузол чистий (незмішаний), якщо належить лише одному класу. Якщо вузол розділяється на лівий вузол l та правий вузол r з частотою входження у групу лівих вузлів $p(l)$ та частотою входження у групу правих вузлів $p(r)$, оцінити рівень змішаності після такого розділення можна за допомогою величини

$$I = i(v) - p(l)i(l) - p(r)i(r). \quad (2.2.13)$$

Чим більша величина I тим більше змішане розділення, тобто нові вершини більш чисті. Отже величину I необхідно максимізувати, що еквівалентно мінімізації виразу

$$p(l)i(l) + p(r)i(r).$$

Якщо відійти від лінійності у пропорціях ймовірності змішаності, можна розглянути квадратичний індекс Джині, в якому більш чистим вузлам відповідають більші вагові коефіцієнти. Визначивши $i(v) = p(G|v)\,p(B|v)$, маємо:

$$G = p(G|v)\,p(B|v) - p(l)\,p(G|l)\,p(B|l) - \\ -p(r)\,p(G|r)\,p(B|r). \qquad (2.2.14)$$

Індекс ентропії. Ще один нелінійний індекс це індекс ентропії, що пов'язаний з кількістю інформації у поділі «хороших» та «поганих» заявників у вузлі:

$$i(v) = -p(G|v)\ln\big(p(G|v)\big) - p(B|v)\ln\big(p(B|v)\big). \qquad (2.2.15)$$

Максимізація напівсуми квадратів. Ця міра випливає з статистичного χ^2-тесту і визначає чи співпадають частки «хороших» заявників у множині, що розділяється, та двох підмножинах, що при цьому утворились. Чим більша статистика χ^2 тим менш схожі частки, тобто більш значна різниця між ними. Якщо $n(l)$ та $n(r)$ – кількості лівих та правих вузлів, задача полягає у максимізації величини

$$Chi = n(l)n(r) - \frac{\big(p(G|l) - p(G|r)\big)^2}{n(l) + n(r)}. \qquad (2.2.16)$$

Метод *k* найближчих сусідів. Одним з найдавніших непараметричних методів класифікації є метод *k* найближчих сусідів. Ідея методу полягає у тому, щоб множину вже класифікованих заявників, кожному з яких відповідає вектор їх атрибутів, розмістити в просторі цих атрибутів. У цьому просторі вводиться метрика для визначення відстані між заявниками. Віднесення нового заявника до певного класу відбувається залежно від властивостей *k* найближчих до нього сусідів. Величина *k* – параметр і регулюється окремо.

Для роботи методу необхідно визначити параметр *k*, частку «хороших» заявників серед *k* сусідів для віднесення нового заявника до класу «хороших» та метрику простору. Величина параметра *k* суттєво залежить від розміру вибірки заявників та в деяких випадках результат при різних *k* значно відрізняється, тому зазвичай підбирається емпіричним шляхом. Як частка «хороших» заявників серед сусідів для класифікації зазвичай береться більшість таких заявників або обирається таким чином, щоб мінімізувати сумарні втрати за хибну класифікацію. Одна найбільш вдала

глобальна метрика для використання в задачах кредитного скорингу це метрика, що є комбінацією евклідової метрики та відстані у напрямку w, який найкращим чином розділяє «хороших» та «поганих» заявників. Вона має такий вигляд:

$$d(x_1, x_2) = \left\{ (x_1 - x_2)^T \left(I + Dw \cdot w^T \right) (x_1 - x_2) \right\}^{\frac{1}{2}}, \qquad (2.2.17)$$

де I – одинична матриця, D – середні втрати у випадку банкрутства заявника.

Варто відмітити, що метод k найближчих сусідів має декілька важливих переваг порівняно з іншими методами, що застосовуються у кредитному скорингу. Зокрема, це можливість динамічно додавати й класифікувати нових заявників, поповнюючи тренувальний набір. Також це можливість змінювати метрику, оскільки правильно підібрана метрика суттєво впливає на якість кінцевого результату.

Мультигрупова дискримінація. Іноді в кредитному скорингу виникає потреба класифікувати заявників на більше ніж 2 групи. Наприклад, необхідно виділити заявників, яким кредитор готовий надати позику, небажаних заявників через банкрутство в минулому та небажаних заявників через недостатній прибуток для кредитора. В такому випадку застосовують методи мультигрупової дискримінації, до переліку яких входить більшість із вищенаведених методів з відповідними модифікаціями. Розглянемо модифікацію підходу прийняття рішень у дискримінантному аналізі.

Нехай $c(i, j)$ – збитки за віднесення заявника j до групи i, а p_j – частка групи j у вибірці. Позначимо як $p(x \mid j)$ ймовірність того, що заявники з групи j мають вектор атрибутів x. Тоді ймовірність того, що заявник з вектором атрибутів x належить до групи j, рівна

$$p(j \mid x) = \frac{p_j p(x \mid j)}{\sum_i p_i p(x \mid i)}. \qquad (2.2.18)$$

Для мінімізації сумарних витрат заявник з вектором атрибутів призначається до групи i якщо

$$\sum_j c(i,j)\, p_j\, p(x \mid j) < \sum_j c(k,j)\, p_j\, p(x \mid j), \quad \forall k,\ k \neq i. \quad (2.2.19)$$

2.2.4 Нестатистичні методи для побудови кредитних скорингових карток

На початкових етапах розвитку кредитного скорингу здебільшого використовувались статистичні методи розв'язання поставлених задач. Однак період інтенсивного розвитку кредитного скорингу співпав з періодом активних досліджень в області методів дослідження операцій, математичного програмування, штучного інтелекту, а згодом і машинного навчання. Задача класифікації, що лежить у основі кредитного скорингу, була сформульована в багатьох різних формах, що дозволяло застосовувати для її розв'язання широкий набір нестатистичних підходів та методів. Наведемо короткий опис таких методів.

Лінійне програмування. Вперше підхід лінійного програмування для розв'язання задач класифікації був використаний у роботі [7], де метою була побудова гіперплощини, що розділяє дві групи. Якщо групи не є лінійно роздільними, проводиться мінімізація суми модулів нев'язок або мінімізація максимуму нев'язки.

Нагадаємо, що задача кредитного скорингу полягає у розділенні множини A – набору всіх комбінацій значень p змінних $X = \left(X_1, ..., X_p\right)$ – на дві множини A_G та A_B «хороших» та «поганих» заявників відповідно. Нехай n – кількість заявників у вибірці, а n_G та n_B – кількість «хороших» та «поганих» заявників відповідно, причому $n = n_G + n_B$. Не обмежуючи загальності припустимо, що «хорошими» заявниками є n_G перших заявників у вибірці. Нехай i-й заявник має вектор атрибутів $\left(x_{i1}, ..., x_{ip}\right)$. Необхідно визначити вектор ваг $\left(\omega_1, ..., \omega_p\right)$ так, щоб зважена сума $\omega_1 X_1 + ... + \omega_p X_p$ перевищувала прохідний бал c для «хороших» заявників та не перевищувала для

«поганих». Для отримання наближеного розв'язку задачі вводиться вектор нев'язок $a=(a_1,...,a_n)$, тоді умови класифікації формулюються так: якщо заявник i «хороший» має виконуватись нерівність $\omega_1 x_{i1}+...+\omega_p x_{ip}\geq c-a_i$, інакше виконується нерівність $\omega_1 x_{i1}+...+\omega_p x_{ip}\leq c-a_i$. Тоді для пошуку вектора ваг $(\omega_1,...,\omega_p)$, що мінімізує суму модулів відхилень, необхідно розв'язати таку задачу лінійного програмування:

$$a_1+a_2+...+a_{n_G+n_B}\to\min \tag{2.2.20}$$

за умов

$$\omega_1 x_{i1}+...+\omega_p x_{ip}\geq c-a_i,\quad i=\overline{1,n_G}, \tag{2.2.21}$$

$$\omega_1 x_{i1}+...+\omega_p x_{ip}\leq c-a_i,\quad i=\overline{1+n_G,n_G+n_B}, \tag{2.2.22}$$

$$a_i\geq 0,\quad i=\overline{1,n_G+n_B}. \tag{2.2.23}$$

Якщо ж мінімізується максимум відхилень, задача має такий вигляд:

$$a\to\min \tag{2.2.24}$$

за умов

$$\omega_1 x_{i1}+...+\omega_p x_{ip}\geq c-a,\quad i=\overline{1,n_G}, \tag{2.2.25}$$

$$\omega_1 x_{i1}+...+\omega_p x_{ip}\leq c-a,\quad i=\overline{1+n_G,n_G+n_B}, \tag{2.2.26}$$

$$a_i\geq 0. \tag{2.2.27}$$

Використання лінійного програмування у кредитному скорингу має як переваги, так і недоліки. Зокрема, цей підхід дозволяє встановити бажане відхилення на етапі розробки скорингових карток. Взаємозв'язки між певними змінними легко встановлюються за допомогою лінійних обмежень, які додаються до системи обмежень задачі. Однак обмеження задачі завжди мають форму рівностей або нестрогих нерівностей, виключаючи використання строгих нерівностей. Це може призводити до отримання тривіальних розв'язків, коли всі елементи вектора ваг та прохідний бал рівні нулю. Для уникнення таких ситуацій прохідний бал встановлюється

ненульовим, однак у такому разі задачу необхідно розв'язати двічі: для додатного та від'ємного значень прохідного балу.

Ще один важливий недолік застосування лінійного програмування це відсутність можливості оцінити статистичну значимість параметрів, що оцінюються. Один із шляхів подолання цієї проблеми – це спосіб оцінки параметрів за допомогою технік статистичного бутстрепу (bootstrap) та методу складного ножа (jackknife). Також одна з переваг регресійного підходу над лінійним програмуванням це можливість почергово вводити у рівняння регресори, починаючи з найбільш значимого, що дозволяє будувати моделі з заданим числом регресорів, які будуть найбільш ефективними в дискримінації. В роботі [8] показано як можна використати підхід складного ножа до лінійного програмування, щоб отримати фіксоване число найбільш ефективних характеристик. Насправді, цей підхід також вимагає багаторазового розв'язання задачі лінійного програмування.

Цілочислове програмування. Ще один підхід розв'язання задачі кредитного скорингу це мінімізація числа неправильних класифікацій або сумарних втрат при неправильній класифікації, якщо величина D (втрати через класифікацію «поганого» заявника як «хорошого») значно відрізняється від величини L (втрати через класифікацію «хорошого» заявника як «поганого»). Така модель теж лінійна, причому певні змінні мають бути цілочисловими, отже маємо задачу цілочислового лінійного програмування:

$$L\left(d_1 + ... + d_{n_G}\right) + D\left(d_{n_{G+1}} + ... + d_{n_{G+B}}\right) \to \min \qquad (2.2.28)$$

за умов

$$\omega_1 x_{i1} + ... + \omega_p x_{ip} \geq c - Md_i, \quad i = \overline{1, n_G}, \qquad (2.2.29)$$

$$\omega_1 x_{i1} + ... + \omega_p x_{ip} \leq c - Md_i, \quad i = \overline{1 + n_G, n_G + n_B}, \qquad (2.2.30)$$

$$0 \leq d_i \leq 1, \quad d_i \text{ – цілочислові.} \qquad (2.2.31)$$

Змінна d_i приймає значення один, якщо заявника i класифіковано неправильно, та нуль в іншому випадку. Для уникнення отримання тривіальних розв'язків, аналогічно до задачі лінійного програмування необхідно ввести умови нормалізації:

$$\sum_{j=1}^{p}\left(s_j^+ + s_j^-\right)=1,\ s_j^+ \geq 0,\ s_j^- \leq 1,\ s_j^+,\ s_j^-\ -\text{цілочислові},\ j=\overline{1,p}\ , \quad (2.2.32)$$

$$-1+2s_j \leq \omega_j \leq 1-2s_j,\quad j=\overline{1,p}\ . \quad (2.2.33)$$

Такі умови аналогічні до умов

$$\sum_{j=1}^{p}\omega_j = 1 \text{ та } c=+1 \text{ або } -1.$$

Описана модель виявилась кращою класифікаційною моделлю, ніж модель лінійного програмування. Зокрема, вона дозволяє усунути тривіальні розв'язки та забезпечити інваріантність ваг, якщо дані є зміщеними – це забезпечується за допомогою нормалізації (2.2.32), (2.2.33). Підхід цілочислового програмування дозволяє вирішити проблему створення оптимальної скорингової картки, використовуючи лише m характеристик. Для цього необхідно додати обмеження

$$\sum_{j=1}^{p}\left(s_j^+ + s_j^-\right)=m,$$

завдяки якому лише m ваг будуть додатними та лише m характеристик будуть ненульовими.

Однак, підхід цілочислового програмування має два суттєвих недоліки. Перший – розв'язання задачі цілочислового програмування потребує значно більше часу, ніж лінійного програмування, тому обсяг вибірки може бути відносно невеликим. Це призводить до малого поширення використання цього підходу в комерційних застосуваннях кредитного скорингу. Другий – полягає у тому, що зазвичай наявно багато оптимальних розв'язків з однаковим числом неправильних класифікацій, але суттєво різною якістю на вихідних вибірках.

Метод опорних векторів (SVM). Важливу роль в сучасному машинному навчанні відіграє метод опорних векторів (МОВ) або SVM (support vector machines), який можна використовувати для розв'язання задач класифікації, кластеризації, регресії та багатьох

інших задач, які до них зводяться. SVM був запропонований в 1963 році [9] та був призначений для лінійного розділення двох множин точок. В 1992 році було запропоновано спосіб будувати нелінійні класифікатори шляхом застосування до максимально розділових гіперплощин так званого ядрового трюку [10].

Нехай $\{(X_i, y_i)\}_{i=1}^{n}$ – тренувальна вибірка, де X_i – p-вимірний вектор атрибутів i-го заявника, $y_i \in \{-1, +1\}$ – клас, до якого цей заявник був віднесений, причому значення -1 відповідає класу «поганих» заявників, а значення $+1$ – класу «хороших» заявників. Завданням методу опорних векторів є визначення оптимальних значень параметрів $\vec{\omega}$ та b так, щоб гіперплощина $\vec{\omega} \cdot \vec{x} - b = 0$ розділяла множину точок X_i, для яких $y_i = 1$, та множину точок X_i, для яких $y_i = -1$, причому відстань від цієї гіперплощини до найближчої точки з кожної множини максимальна.

Якщо дані є лінійно роздільними, будуються дві гіперплощини $\vec{\omega} \cdot \vec{x} - b = 1$ та $\vec{\omega} \cdot \vec{x} - b = -1$, відстань між якими необхідно максимізувати: $2/\|\vec{\omega}\| \to \max$ (або $\|\vec{\omega}\| \to \min$). Розв'язком оптимізаційної задачі

$$\|\vec{\omega}\| \to \min, \tag{2.2.34}$$

$$y_i\left(\vec{\omega} \cdot \vec{X}_i - b\right) \geq 1, \ \forall i = \overline{1,n}, \tag{2.2.35}$$

є вектор $\vec{\omega}$ та скаляр b, які визначають класифікатор $\vec{X} \mapsto \operatorname{sgn}\left(\vec{\omega} \cdot \vec{X} - b\right)$. Такий метод опорних векторів має назву МОВ з жорстким розділенням (hard-margin SVM). Для розширення методу опорних векторів на випадок, коли дані не є лінійно роздільними, вводиться так звана завісна функція втрат (hinge loss function)

$$\ell(y_i) = \max\left(0, 1 - y_i\left(\vec{\omega} \cdot \vec{X}_i - b\right)\right),$$

яка рівна 0, якщо обмеження (2.35) задовольняється, тобто положення точки $\vec{X}_i$ встановлено правильно. Інакше значення функції $\ell(y_i)$ є

пропорційним до відстані від розділення до неправильно класифікованої точки $\vec{X}_i$. Оптимізаційна задача полягає в мінімізації функції

$$F\left(\vec{\omega},b\right)=\left[\frac{1}{n}\sum_{i=1}^{n}\max\left(0,1-y_i\left(\vec{\omega}\cdot\vec{X}_i-b\right)\right)\right]+\lambda\left\|\vec{\omega}\right\|^2, \quad (2.2.36)$$

де параметр λ визначає співвідношення між розміром розділення та правильністю класифікації $\vec{X}_i$. Такий підхід має назву МОВ з м'яким розділенням (soft-margin SVM) та при досить малих значеннях λ поводиться аналогічно до МОВ з жорстким розділенням, якщо дані є лінійно роздільними, інакше – відбувається процес навчання.

В задачах кредитного скорингу дані часто не є лінійно роздільними. В такому випадку аналогічним чином будується лінійний класифікатор, в якому скалярний добуток замінюється нелінійною ядровою функцією, що дозволяє узгоджувати максимально розділову гіперплощину в перетвореному просторі ознак. До найбільш поширених ядерних функцій належать поліноміальна однорідна $k\left(\vec{X}_i,\vec{X}_j\right)=\left(\vec{X}_i\cdot\vec{X}_j\right)^d$ та неоднорідна $k\left(\vec{X}_i,\vec{X}_j\right)=\left(\vec{X}_i\cdot\vec{X}_j+1\right)^d$, сигмоїдна $k\left(\vec{X}_i,\vec{X}_j\right)=\tanh\left(a\vec{X}_i\cdot\vec{X}_j+b\right)$ для деяких $a>0$ та $b<0$ тощо.

Ефективність використання методу опорних векторів для розв'язання задач кредитного скорингу виявилась на одному рівні з іншими методами на кшталт лінійної регресії, лінійного дискримінантного аналізу, генетичних алгоритмів тощо. В роботах [11–13] наведено результати застосування МОВ для задач кредитного скорингу.

Нейронні мережі. Один з найпоширеніших інструментів у сучасному штучному інтелекті це нейронні мережі, які дозволяють розв'язувати широкий спектр різних задач. Найпростіша одношарова нейронна мережа складається з вхідного шару нейронів (набору входів, через які до мережі передаються дані), сумуючої функції, яка підсумовує значення входів з певними вагами, та активаційної

функції, яка забезпечує вихід мережі. В термінах кредитного скорингу, через вхідні нейрони до мережі передаються дані про заявника, а на виході мережі отримується результат класифікації заявника як «хорошого» або «поганого». Алгебраїчно модель такої мережі, що має назву перцептрон Розентблата, можна виразити так:

$$u_k = \omega_{k0}x_0 + \omega_{k1}x_1 + ... + \omega_{kp}x_p = \sum_{q=0}^{p} \omega_{kq}x_q \ , \qquad (2.2.37)$$

$$y_k = F\left(u_k\right). \qquad (2.2.38)$$

Тут $\left(x_1,...,x_p\right)$ – набір вхідних даних (характеристик), x_0 – відхилення, $\left(\omega_{k0},...,\omega_{kp}\right)$ – ваги, F – функція активації, y_k – вихід мережі, k – індекс нейрону в наступному шарі мережі (для одношарової мережі $k=1$).

Перцептрон дозволяє класифікувати дані тільки у тому випадку, коли вони лінійно роздільні. Для класифікації нелінійно роздільних даних використовуються багатошарові перцептрони з нелінійними функціями активації. Така мережа складається з вхідного та вихідного шарів, а також набору прихованих шарів. Кожен нейрон першого прихованого шару має набір ваг, які разом із значеннями вхідних нейронів використовуються у сумуючій функції. Значення сумуючої функції передається до функції активації, значення якої слугує входом для наступного шару мережі. Таким чином, набір вхідних даних, що надходить до мережі, проходить через низку нейронів, що мають різні ваги та активаційні функції, формуючи набір вихідних даних. Алгебраїчно модель багатошарової нейронної мережі можна представити так:

$$y_k = F_1\left(\sum_{q=0}^{p} \omega_{kq}x_q\right), \qquad (2.2.39)$$

де індекс 1 вказує, що це перший шар після вхідного. Величини y_k – вихідні значення першого прихованого шару. Оскільки вихідні значення одного шару є вхідними значеннями для наступного, вихід мережі можна записати так:

$$z_v = F_2\left(\sum_{k=1}^{r} K_{vk} y_k\right) = F_2\left(\sum_{k=1}^{r} K_{vk}\left(F_1\left(\sum_{q=0}^{p} \omega_{kq} x_q\right)\right)\right), \qquad (2.2.40)$$

де z_v – вихід нейрона v вихідного шару, F_2 – функція активації вихідного шару, K_{vk} – матриця ваг між прихованим та вихідним шарами.

Для використання мережі як класифікатора необхідно обчислити ваги всієї мережі. Один з найбільш поширених алгоритмів для цього процесу це метод оберненого розповсюдження помилки (back-propagation algorithm), який є методом градієнтного спуску. Він полягає у послідовному коректуванні ваг мережі шляхом надання їй правильно класифікованих прикладів, при цьому мінімізується певна функція похибок.

Для досягнення максимальної ефективності роботи нейронної мережі необхідно правильно визначити архітектуру мережі, а саме кількість прихованих шарів та кількість нейронів у цих шарах та функцію похибок. Число прихованих шарів зазвичай обирають рівним 2. Перший шар дозволяє отримати значення вище або нижче прохідного балу в опуклій області вхідних змінних, другий – комбінувати ці опуклі області, що може дати неопуклі або повністю розділені області. Число нейронів у прихованих шарах зазвичай підбирається з використанням евристичних процедур.

Використання нейронних мереж дозволяє проводити класифікацію для довільної кількості класів шляхом регулювання кількості вихідних нейронів мережі. Відомо, що багатошаровий перцептрон, натренований з використанням методу оберненого розповсюдження помилки та відповідною функцією похибок скінченою кількістю незалежних прикладів та однаково розподіленими вхідними даними, асимптотично збігається до апроксимації апостеріорних ймовірностей належності до класів. Отже заявник g належить до групи C_g, якщо вихідне значення вихідного шару, на якому ця група була натренована, $F_g(x)$, є більшим, ніж вихідне значення вихідного шару, на якому була натренована довільна

інша група – $F_h(x)$, тобто якщо $F_g(x) > F_h(x)$, $g \neq h$. Функцію похибок можна визначити таким чином. Ймовірності отримання вихідних значень з вектором вхідних даних $x(t)$ є y_g, розподіл яких – $P(o(t) \mid x(t)) = \prod_{g=1}^{Z} \left(y_g^t\right)^{o_g^t}$. Звідси отримуємо критерій відносної ентропії: $E_2 = -\sum_t \sum_{g=1}^{Z} O_g^t \ln y_g^t$. Оскільки значення y_v інтерпретуються як ймовірності, для забезпечення умов $0 \le y_{vg} \le 1$ та $\sum_{g=1}^{Z} y_{vg} = 1$ використовується активаційна функція softmax:

$$y_g = \frac{e^{u_g}}{\sum_{g=1}^{Z} e^{u_g}}.$$

Нейронні мережі знайшли своє застосування у багатьох прикладних областях, у тому числі в фінансовому та кредитному секторах для прогнозування банкрутства, виявлення махінацій з кредитними картками, аналізу застав, ціноутворення опціонів тощо [14–16].

Градієнтний бустинг. Багато ефективних методів класифікації належать до області машинного навчання. Одним з найбільш широковживаних методів є градієнтний бустинг (gradient boosting). Його мета полягає у створенні ансамблю «слабких» класифікаторів для підвищення їх ефективності. Як класифікатори зазвичай обирають дерева прийняття рішень, тому утворений ансамбль має назву градієнтний бустинг над деревами прийняття рішень (gradient boosted decision tree).

Нехай $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ – тренувальна вибірка, де $\mathbf{x}_i$ – вектор характеристик *i*-го заявника, $y_i \in \{0,1\}$ – результат класифікації цього заявника. Для класифікації заявників будується функція $F(\mathbf{x}_i)$ така,

що мінімізує функцію втрат $L\left(y_i, F\left(\mathbf{x}_i\right)\right)$:

$$F^* = \arg\min_F \sum_{i=1}^{N} L\left(y_i, F\left(\mathbf{x}_i\right)\right). \quad (2.2.41)$$

Функція $F\left(\mathbf{x}\right)$ – адитивна та будується ітераційно:

$$F\left(\mathbf{x}\right) = \sum_{t=1}^{T} f_t\left(\mathbf{x}\right), \quad (2.2.42)$$

де кожна функція f – дерево прийняття рішень, T – кількість ітерацій. На ітерації t функція f_t реалізує подальшу мінімізацію сукупних втрат ансамблю $\left\{f_j\right\}_{j=1}^{t-1}$.

Згодом після відкриття градієнтного бустингу була запропонована його модифікація, що використовує ідею бутстреп-агрегації Бреймана. У новому методі доля навчальних прикладів для кожного дерева знижується, внаслідок чого навчання значно прискорюється, а також більш високе зміщення замінюється низькою дисперсією. Такий метод має назву стохастичний градієнтний бустинг (stochastic gradient boosting).

Генетичні алгоритми. Ще один досить ефективний інструмент для розв'язання задач кредитного скорингу це генетичні алгоритми. Їх мета полягає у систематичному пошуку в множині можливих розв'язків задачі таким чином, що якість розв'язків прямо пропорційна ймовірності залишитись у цій множині. Одна з можливих моделей, яка може бути використана для класифікації заявників, така:

$$f\left(x_i\right) = a_1 x_{i1}^{b_1} + a_2 x_{i2}^{b_2} + ... + a_p x_{ip}^{b_p} + c, \quad (2.2.43)$$

де $a_1, a_2, ..., a_p, b_1, b_2, ..., b_p, c$ – параметри, які необхідно знайти, $x_{i1}, ..., x_{ip}$ – характеристики заявника i. Залежно від значення функції $f\left(x_i\right)$ (додатного чи від'ємного) заявник класифікується як «хороший» або «поганий» відповідно. Для кожного параметра встановлюються верхні та нижні границі.

Робота генетичного алгоритму здійснюється у декілька етапів.

Спочатку визначається множина вихідних розв'язків задачі (вихідна популяція) випадковим чином за допомогою верхніх та нижніх границь для параметрів. На другому етапі для кожного розв'язку з цієї множини обчислюється його нормалізована продуктивність p_j з використанням функції придатності (fitness function). В задачах кредитного скорингу така функція, наприклад, може обчислювати кількість правильно класифікованих заявників. Використовуючи величини p_j як ймовірності бути обраним, формується проміжна вибірка розв'язків (проміжна популяція) обсягу n_p. На третьому етапі виконується генерація нових розв'язків на основі наявних у проміжній популяція з використанням функцій схрещування (crossover) та мутації (mutation). Розв'язки зазвичай кодуються за допомогою нуля та одиниці, або набору цифр від нуля до дев'яти. Кожен розв'язок це послідовний набір груп цифр; перша цифра – індикатор і вказує на те володіє чи не володіє заявник певною характеристикою. Решта цифр групи – значення характеристики або її інтервал. Наприклад, розв'язок 11018 означає, що заявник має домашній телефон (перша цифра рівна 1) та не проживає за вказаною адресою від 1 до 8 років (третя цифра рівна 0, дві останні – 1 та 8). Функція мутації дозволяє отримати новий розв'язок шляхом взаємозаміни фіксованих частин двох обраних розв'язків, наприклад, перших n цифр. Функція мутації змінює певні частини розв'язку, наприклад, індикатори з 0 на 1. Обраний розв'язок разом з результатами роботи функцій схрещування та мутації формують нову популяцію. Етапи 2 і 3 повторюються фіксовану кількість разів.

Генетичні алгоритми та їх розширення генетичне програмування – одні з відносно недавніх інструментів, що знайшли своє застосування у кредитному скорингу [17, 18] та прогнозуванні банкрутств [19, 20].

Експертні системи набули широкого поширення наприкінці XX століття. Це набір процесів, метою яких є емуляція роботи експерта у певній області. Зазвичай експертна система складається з бази знань (правил) та набору фактів, з яких за допомогою генератора висновків

отримуються рекомендації до дій. Правила зазвичай мають форму «якщо-тоді», наприклад, «якщо річні виплати перевищують 50 % річних надходжень, тоді кредит не буде погашено». Для побудови бази знань проводиться робота з фахівцем в області, роботу якого система має імітувати. Також для отримання таких правил використовуються нейронні мережі, які після тренування за вхідними даними (характеристиками заявника) видають певне рішення. Однак такі рішення не інтерпретовані, тому одне із завдань експерта їх обґрунтування та пояснення.

Основні переваги експертних систем – це просте розширення бази знань шляхом додавання нових правил та інтерпретація рішень, що видаються системами. Наприклад, побудована в роботі [21] експертна система CLUES для прийняття рішень щодо страхування іпотечної позики видала рішення, практично всі з яких були схвалені страховиками. Система містить близько 1000 правил, які оцінювали кожного заявника на предмет кожного з трьох типів аналізу, які зазвичай проводяться страховиками: аналіз платоспроможності позичальника, аналіз його спроможності до погашення позики та розгляд оціночного висновку. Застосування експертних систем у кредитному скорингу та їх порівняння з іншими методами висвітлено в [22, 23].

Мультикритеріальний скоринг з використанням методів BWM та TOPSIS. Для оцінки фінансових та нефінансових показників, які впливають на платоспроможність малих та середніх підприємств, за умови дефіциту фінансових даних досить ефективним інструментом виявились моделі мультикритеріального скорингу, побудовані з використанням методу BWM та підходу TOPSIS [24].

Мультикритеріальний скоринг базується на багатокритеріальних моделях прийняття рішень, які дозволяють отримати компромісне рішення, зберігаючи особу, що приймає рішення, в центрі системи. Його мета – визначення придатних організацій для надання кредитів. Для оцінки ваг критеріїв використовується метод типу «кращий-гірший» (best-worst method (BWM)), а для обчислення кредитних оцінок кожної організації застосовується техніка впорядкування

пріоритетів за близькістю до ідеального рішення (technique for order of preference by similarity to ideal solution (TOPSIS)).

Метод BWM, запропонований в роботі [25], включає декілька етапів. На першому етапі визначається множина критеріїв $C_1,...,C_n$, серед яких експертами обирається найкращий та найгірший критерії. Після чого будуються вектори $A_B = \left(a_{B1},...,a_{Bn}\right)^T$ та $A_W = \left(a_{1W},...,a_{nW}\right)^T$ таким чином: елемент a_{Bi} є оцінкою того, наскільки найкращий критерій переважає i-й за шкалою 1–9; елемент a_{jW} є оцінкою того, наскільки j-й критерій переважає найгірший. Вектор оптимальних ваг критеріїв $\left(w_1^*,...,w_n^*\right)$ обчислюється шляхом розв'язання такої мінімаксної задачі:

$$\min \max_j \left\{ \left| \frac{w_B}{w_j} - a_{Bj} \right|, \left| \frac{w_j}{w_W} - a_{jW} \right| \right\}, \tag{2.2.44}$$

$$\sum_{j=1}^{n} w_j = 1,\ w_j \geq 0,\ j = \overline{1,n}, \tag{2.2.45}$$

яку можна переформулювати так:

$$\xi^l \to \min, \tag{2.2.46}$$

$$\left| \frac{w_B}{w_j} - a_{Bj} \right| \leq \xi^l,\ \forall j = \overline{1,n}, \tag{2.2.47}$$

$$\left| \frac{w_j}{w_W} - a_{jW} \right| \leq \xi^l,\ \forall j = \overline{1,n}, \tag{2.2.48}$$

$$\sum_{j=1}^{n} w_j = 1,\ w_j \geq 0,\ j = \overline{1,n}, \tag{2.2.49}$$

де ξ^l – конзистентність прийняття рішень. Розв'язуючи лінійну модель (2.2.46) – (2.2.49) отримуємо оптимальні ваги $\left(w_1^*,...,w_n^*\right)$ та конзистентність ξ^l. Також для перевірки рівня конзистентності

попарних порівнянь критеріїв обчислюється відношення конзистентності $CR = \frac{\xi^l}{CI}$, де CI – табличне значення індексу конзистентності [25].

Підхід TOPSIS, запропонований в роботі [26], забезпечує для особи, що приймає рішення, можливість обрати найкращу з наявних альтернатив. Результати використання цього підходу після застосування багатокритеріальних моделей прийняття рішень значно кращі, ніж при використанні самих моделей [27].

Робота підходу TOPSIS описується такою процедурою. Нехай проблема прийняття рішень включає альтернативи $A_1,...,A_m$ та критерії $C_1,...,C_n$. Для оцінки альтернатив за критеріями будується матриця $A_{ij} = \left(a_{ij}\right)_{i,j=1}^{m,n}$, в якій рядки A_i відповідають альтернативам, стовпці C_j – критеріям, а елемент a_{ij} є оцінкою i-ї альтернативи j-м критерієм. Потім будується нова матриця $V_{ij} = \left(w_i r_{ij}\right)_{i,j=1}^{m,n}$, де $r_{ij} = a_{ij} \Big/ \sqrt{\sum_{i=1}^{m} a_{ij}^2}$, $i = \overline{1,m}$, $j = \overline{1,n}$ – нормалізовані елементи матриці A_{ij}, а w_i, $i = \overline{1,m}$ – оптимальні ваги критеріїв, обчислені за допомогою методу BWM. На наступному етапі визначаються позитивне ідеальне рішення A^* та негативне ідеальне рішення A^-:

$$A^* = \left\{ \left[\max_{i=1,m,\ j\in J_+} v_{ij} \right], \left[\min_{i=1,m,\ j\in J_-} v_{ij} \right] \right\} = \left\{ v_1^*,...,v_n^* \right\}, \qquad (2.2.50)$$

$$A^- = \left\{ \left[\min_{i=1,m,\ j\in J_+} v_{ij} \right], \left[\max_{i=1,m,\ j\in J_-} v_{ij} \right] \right\} = \left\{ v_1^*,...,v_n^* \right\}, \qquad (2.2.51)$$

де J_+ та J_- – індекси $j = \overline{1,n}$, асоційовані з позитивними та негативними критеріями відповідно. Після чого обчислюються

відносні близькості до ідеального рішення $C_i^* = \frac{S_i^-}{S_i^- + S_i^*}$, де $S_i^- = \sqrt{\sum_{j=1}^{m}\left(v_{ij} - v_j^-\right)^2}$, $S_i^* = \sqrt{\sum_{j=1}^{m}\left(v_{ij} - v_j^*\right)^2}$, $i = \overline{1,m}$ – евклідові відстані альтернатив між негативним та позитивним ідеальними рішеннями відповідно. Отримані величини C_i^* можна використати для впорядкування альтернатив.

Перевагами мультикритеріального скорингу, зокрема з використанням методів BWM та TOPSIS є можливість враховувати як числові, так і нефінансові показники, точність прогнозів та низька обчислювальна вартість методу. Підхід не вимагає ніяких припущень, на відміну від статистичних методів, що робить підхід більш гнучким. Застосування мультикритеріального скорингу для задач кредитного скорингу є досить обширним [28–31].

2.2.5 Порівняння методів

Всі вищерозглянуті методи якісно різні, тому визначити найкращий з них не вдається. Ефективність роботи кожного з методів залежить від низки факторів, які визначають доцільність їх застосування у кожному конкретному випадку. Зокрема, регресійний підхід дозволяє оцінити значимість кожного фактора за допомогою статистичних тестів, а кореляційний аналіз – значимість впливу різних факторів, що в результаті дозволяє послідовно побудувати стійку та надійну модель. Підхід лінійного та цілочислового програмування дає можливість легко додавати до моделі нові обмеження, встановлені позикодавцями, а також розв’язувати задачу з великою кількістю характеристик заявників. Перевага нейронних мереж та класифікаційних дерев полягає у тому, що вони автоматично виявляють та опрацьовують взаємодії між характеристиками, що дозволяє виділяти різні групи заявників та створювати окремі скорингові картки для них. Метод найближчих сусідів та генетичні

алгоритми дають можливість будувати скорингові картки, які можна динамічно оновлювати, додаючи нові характеристики та видаляючи старі, які більше не впливають на результат.

Важливий фактор, що впливає на вибір того чи іншого алгоритму для розв'язання задач кредитного скорингу – пріоритети позичальника. Комерційна складова іноді спонукає знизити точність, але підвищити простоту системи. Однак у випадку, коли мова йде про великий обсяг множини заявників або їх характеристик, доцільним рішенням це покращення обчислювальних можливостей позикодавця та вибір оптимізаційних методів для розв'язання задач кредитного скорингу.

2.2.6 Висновки

Кредитний скоринг є важливим інструментом в області фінансів та банкінгу. Ефективність розв'язання задач кредитного скорингу безпосередньо впливає на якість банківських послуг, що надаються клієнтам, оцінку ризиків фінансових операцій, надійність функціонування кредитної системи тощо. Правильна оцінка ризиків при наданні позик дозволяє мінімізувати втрати й стабілізувати фінансову систему загалом. Методи і процеси для оцінки різного роду ризиків активно розвиваються, вдосконалюються та застосовуються для все більшої кількості задач оцінки ризиків, зокрема фінансових, енергетичних, медичних і багатьох інших. Розробка, аналіз та застосування цих методів проводиться, зокрема в публікаціях [32–35].

В підрозділі розглянуто основні статистичні та нестатистичні методи, які використовуються для розв'язання задач кредитного скорингу. Показано, що у випадку, коли пріоритетом є оцінка факторів моделі, їхньої значимості та кореляції, доцільно використовувати статистичні методи, зокрема регресійний та дискримінантний аналіз, SVM, генетичні алгоритми тощо. Якщо ж об'єм вхідних даних є значним, що є дуже типовою ситуацією в теперішній час, доцільно застосовувати методи лінійного та цілочислового програмування, що дозволяє скористатись ефективними методами розв'язання цих задач

великих розмірів. Для ситуації з багаторазовим додаванням та видаленням конкретних характеристик моделі для досягнення її максимальної ефективності варто застосовувати метод k найближчих сусідів та генетичні алгоритми.

Список літератури

1. Lyn C. Thomas, David B. Edelman, Jonathan N. Crook. Credit Scoring and its Applications. SIAM Monographs on Mathematical Modeling and Computation. Philadelphia. 2002. 243 p. https://doi.org/10.1137/1.9780898718317
2. Sarlija N., Bensic M., Bohacek Z. Multinomial Model in Consumer Credit Scoring. 10th International Conference on Operational Research. Trogir: Croatia. 2004.
3. Abdou H., Pointon J. Credit scoring and decision-making in Egyptian public sector banks. International Journal of Managerial Finance. 2009. Vol 5. N 4. P. 391–406. https://doi.org/10.1108/17439130910987549
4. Abdou H., Pointon J., El Masry A. Neural nets versus conventional techniques in credit scoring in Egyptian banking. Expert Systems with Applications. 2008. Vol 35. N 3. P. 1275–1292. https://doi.org/10.1016/j.eswa.2007.08.030
5. Baesens B., Gestel T.V., Viaene S., Stepanova M., Suykens J., Vanthienen J. Benchmarking State-of-the-Art Classification Algorithms for Credit Scoring. Journal of the Operational Research Society. 2003. Vol 54. N 6. P. 627–635. https://doi.org/10.1057/palgrave.jors.2601545
6. Юринець З.В., Юринець Р.В., Кунанець Н.Е., Мищишин І.Р. Регресійна модель оцінювання платоспроможності клієнта та банківських ризиків у процесі кредитування. Соціально-економічні проблеми сучасного періоду України. 2019. № 138(4). С. 69–73. https://doi.org/10.36818/2071-4653-2019-4-11
7. Mangasarian O.L. Linear and nonlinear separation of patterns by

linear programming. Oper. Res. 1965. Vol 13. P. 444–452. https://doi.org/10.1287/opre.13.3.444
8. Nath R., Jones T.W. A variable selection criterion in the linear programming approaches to discriminant analysis. Decision Sci. 1988. Vol 19. P. 554–563. https://doi.org/10.1111/j.1540-5915.1988.tb00286.x
9. Вапник В.Н., Червоненкис А.Я. Об одном классе алгоритмов обучения распознаванию образов. Автомат. и телемех. 1964. № 25(6). С. 937–945.
10. Boser B.E., Guyon I.M., Vapnik V.N. A training algorithm for optimal margin classifiers. Proceedings of the fifth annual workshop on Computational learning theory (COLT '92). 1992. P. 144.
11. Bellotti T., Crook J. Support vector machines for credit scoring and discovery of significant features. Expert Systems with Applications. 2009. Vol 36, Issue 2, Part 2. Pp. 3302–3308.
12. Wang Q., Lai K.K., Niu D. Green Credit Scoring System and Its Risk Assessment Model with Support Vector Machine. Fourth International Joint Conference on Computational Sciences and Optimization. 2011. Pp. 284–287.
13. Goh R.Y., Lee L.S. Credit Scoring: A Review on Support Vector Machines and Metaheuristic Approaches. Advances in Operations Research. 2019. Vol 2019. 30 p.
14. Gately E. Neural Networks for Financial Forecasting: Top Techniques for Designing and Applying the Latest Trading Systems. New York: John Wiley & Sons, Inc. 1996.
15. Великоіваненко Г.І., Савіна С.С., Колечко Д.В., Бень В.П. Побудова ансамблів моделей кредитного скорингу. Журн. Нейро-нечіткі технології моделювання в економіці. 2018. Т. 7. С. 34–77. https://www.doi.org/10.33111/nfmte.2018.034/
16. Zekic-Susac M., Sarlija N., Bensic M. Small Business Credit Scoring: A Comparison of Logistic Regression, Neural Networks, and Decision Tree Models. 26th International Conference on Information Technology Interfaces. Croatia. 2004.

17. Huang J., Tzeng G., Ong C. Two-stage genetic programming (2SGP) for the credit scoring model. Applied Mathematics and Computation. 2006. Vol 174. N 2. P. 1039–1053. https://doi.org/10.1016/j.amc.2005.05.027
18. Huang C., Chen M., Wang C. Credit scoring with a data mining approach based on support vector machines. Expert Systems with Applications. 2007. Vol 33. N 4. P. 847–856. https://doi.org/10.1016/j.eswa.2006.07.007
19. Etemadi H., Rostamy A., Dehkordi H. A genetic programming model for bankruptcy prediction: Empirical evidence from Iran. Expert Systems with Applications. 2009. Vol 36. N 2/2. P. 3199–3207. https://doi.org/10.1016/j.eswa.2008.01.012
20. McKee T., Lensberg T. Genetic programming and rough sets: A hybrid approach to bankruptcy classification. European Journal of Operational Research. 2002. Vol 138. N 2. P. 436–451. https://doi.org/10.1016/S0377-2217(01)00130-8
21. Talebzadeh H., Mandutianu S., Winner C. Countrywide loan underwriting expert system. In: Proceedings of the Sixth Innovative Applications of Artificial Intelligence Conference. AAAI Press, Menlo Park, CA. 1994. https://doi.org/10.1609/aimag.v16i1.1123
22. Ben-David A., Frank E. Accuracy of machine learning models versus “hand crafted” expert systems – a credit scoring case study. Expert Systems with Applications. 2009. Vol 36. N 3/1. P. 5264–5271. https://doi.org/10.1016/j.eswa.2008.06.071
23. Kumra R., Stein R., Assersohn I. Assessing a knowledge-based approach to commercial loan underwriting. Expert Systems with Applications. 2006. Vol 30. N 3. P. 507–518.
24. Roy, P.K., Shaw, K. A multicriteria credit scoring model for SMEs using hybrid BWM and TOPSIS. Financ Innov. 2021. 7(77).
25. Rezaei J. Best-worst multi-criteria decision-making method. Omega. 2015. Vol 53. Pp. 49–57.
26. Hwang C.L., Yoon K. Multiple attribute decision making methods and applications a state-of-the-art survey. In: Lecture notes in

economics and mathematical systems. 1981. Vol 186. 269 p.
27. Hsieh L-F., Chin J-B., Wu M-C. Performance evaluation for university electronic libraries in Taiwan. Eletron Library. 2006. 24(2). Pp. 212–224.
28. Gutiérrez-Nieto B., Serrano-Cinca C., Camón-Cala J. A credit score system for socially responsible lending. J Bus Ethics. 2016. 133(4). Pp. 691–701.
29. IÇ Y.T., Yurdakul M. Development of a quick credibility scoring decision support system using fuzzy TOPSIS. Expert Syst Appl. 2010. 37(1). Pp. 567–574.
30. Yang C-C., Ou S-L., Hsu L-C. A hybrid multi-criteria decision-making model for evaluating companies' green credit rating. Sustainability. 2019. 11(6). 1506.
31. Roy P.K., Shaw K. A credit scoring model for SMEs using AHP and TOPSIS. Int J Finance Econ. 2021. Vol 28, Issue 1. Pp. 372–391.
32. Alexei A. Gaivoronski, Pavel S. Knopov and Volodymyr A. Zaslavskyi (eds.) Modern Optimization Methods for Decision Making Under Risk and Uncertainty. CRC Press, Taylor & Francis Group. 2023. 380 p. (is to be published on October, 6, 2023)
33. Ermolieva T., Havlik P., Frank S., Kahil T., Balkovic J., Skalsky R., Ermoliev Y., Knopov P.S., Borodina O.M., Gorbachuk V.M. A Risk-Informed Decision-Making Framework for Climate Change Adaptation through Robust Land Use and Irrigation Planning. Sustainability. 2022. Vol 14(3). P. 1430.
34. Ermolieva T., Ermoliev Y., Zagorodniy A., Bogdanov V., Borodina O., Havlik P., Komendantova N., Knopov P., Gorbachuk V., Zaslavskyi V. Artificial Intelligence, Machine Learning, and Intelligent Decision Support Systems: Iterative "Learning" SQG-based procedures for Distributed Models' Linkage. Artificial Intelligence. 2022. Vol 27 (AI.2022.27). Pp. 92-97.
35. Knopov P.S. Optimization and Identification of Stochastic Systems. Cybernetics and Systems Analysis. 2023. Vol 59(3). P. 375-384.

2.3 МЕТОДИ ОПТИМІЗАЦІЇ РІШЕНЬ В ЗЕЛЕНИХ ЛАНЦЮГАХ ПОСТАЧАННЯ

М. Ю. Григорак

Анотація. Для мінімізації обсягів викидів парникових газів в ланцюгах/мережах постачання запропоновано моделі мережевого потоку з мінімальною вартістю (Minimum Cost Network Flow Problem), які зводяться до задач лінійного програмування. Обґрунтовано використання методів і моделей DEA, що розглядають залежність ефективності/продуктивності ланцюгів постачання між бажаними і небажаними результатами на кожному з етапів доставки сировини від постачальників до виробників продукції, а від них – до кінцевих споживачів. Небажані фактори пов'язані з вимірюванням шкідливого впливу товарного руху на довкілля.

Abstract. Models of minimum cost network flow, which are reduced to linear programming problems, are proposed for minimizing greenhouse gas emissions in supply chains/networks. The use of DEA methods and models that consider the interdependence of efficiency/productivity in supply chains between desirable and undesirable outcomes at each stage of raw material delivery from suppliers to manufacturers, and from them to end consumers, is justified. Undesirable factors are associated with measuring the adverse environmental impact of the movement of goods.

2.3.1 Вступ

Зелені ланцюги постачання (green supply chains) – це підхід до управління ланцюгами постачання, спрямований на зменшення шкідливого впливу на довкілля та забезпечення сталого розвитку. У таких ланцюгах постачання компанії використовують екологічно чисті матеріали, ефективні методи транспортування та утилізації відходів, впроваджують заходи для зменшення викидів шкідливих речовин та сприяють екологічній ефективності в усіх етапах життєвого циклу

товару. Для підвищення ефективності і продуктивності ланцюгів постачання з урахуванням екологічного сліду його учасників активно використовують математичні методи та моделі, які в сукупності дозволяють перерозподілити ризики шкідливого впливу між різними учасниками та знайти кращі управлінські рішення. За допомогою оптимізаційних алгоритмів розв'язують різноманітні задачі математичного програмування з метою пошуку найкращих практик розподілу ресурсів та зменшення викидів.

За даними Глобального вуглецевого бюджету, зростання викидів у 2022 році прискорило підвищення концентрації парникових газів в атмосфері [1]. На сьогоднішній день більшість компаній зосереджуються на скороченні прямих викидів, пов’язаних з операційною діяльністю або з використанням ними електроенергії, тепла та пари. Непрямі викиди пов’язані з рухом товару між ланками ланцюгів постачання від добування сировини до кінцевого споживача. У більшості секторів економіки ці викиди складають понад 80 % викидів парникових газів і більше 90 % впливу на повітря, землю, воду, біорізноманіття [2].

Проблема зменшення викидів парникових газів, які є ключовими факторами глобального потепління, є важливою компонентою стратегічного управління ризиками в ланцюгах/мережах постачання та цінним інструментом для проактивного вирішення конфліктних ситуацій в ланцюгах створення цінності товарів чи послуг. Впровадження бізнес-моделей з низьким рівнем викидів вуглецю може каталізувати розвиток інноваційних підходів до оптимізації логістичних мереж, розташування місць виробництва продукції, розподільчих центрів, екодизайну товарів та їх зберігання, а також залучення інвестицій в екологічні проекти. Сучасні інформаційно-комунікаційні технології дозволяють визначити сфери, де цілі бізнесу та екологічні цілі збігаються, ефективно використовувати наявну цифрову інфраструктуру для відстеження виробничої діяльності та транспортування в майже в реальному часі, взаємодіяти з постачальниками та оцінювати їхній прогрес у досягненні цілей щодо зменшення викидів.

Екологічні аспекти управління ланцюгами постачання є об'єктом дослідження багатьох вчених з різних країн. В наукових працях [3–6] досліджено еволюцію формування концепції «зеленого управління ланцюгом поставок» (GSCM), яка з'явилася на початку 1990-х років як відповідь на зростаючі проблеми навколишнього середовища. Автори [7] стверджують, що екологічні або «зелені» ланцюжки поставок прагнуть створити позитивний вплив на навколишнє середовище, сприяючи безпечному та відповідальному повторному використанню, переробці та утилізації матеріалів.

Розширене трактування GSCM пов'язують з «інтеграцію плану закупівель компанії з природоохоронною діяльністю в ланцюгах постачання для покращення екологічних показників постачальника та клієнтів» [8–10]. Сфера застосування концепції GSCM в літературі варіюється від вибору та постачання матеріалів, процесів виробництва, дизайну продукту, доставки кінцевого продукту кінцевим користувачам та управління виходом продукту після його терміну корисного використання [11, 12]. Зелене виробництво мінімізує відходи та забруднення під час виробничої діяльності. Однак, як зазначають автори [13], взаємозв'язок між екологічними та економічними показниками часто носить суперечливий характер. У багатьох випадках підприємства часто хочуть максимізувати прибуток і мінімізувати викиди вуглецю. Однак максимізація прибутку за рахунок розширення діяльності в ланцюгах поставок часто призводить до збільшення викидів вуглецю. Підприємствам часто важко зрозуміти прямий зв'язок між впровадженням екологічного управління ланцюгом поставок і подальшим підвищенням продуктивності в операційній, економічній або екологічній сферах. На сьогоднішній день низка оглядів літератури підкреслює необхідність врахування викидів парникових газів при проектуванні та плануванні зелених ланцюгів постачання [14, 15] припускає, що вираз «ланцюг постачання» можна було б замінити більш відповідним виразом, таким як «мережа постачання». Ця нова філософія мережі постачання, яка також називається «Зелена мережа постачання» (GSN, Green supply net), є новою тенденцією, яка змушує компанії виробляти та

розвиватися, поважаючи навколишнє середовище та створюючи більш стійку мережу постачання. Автори [16] привертають увагу до того факту, що якщо різні компанії працюють разом, щоб підтримувати задоволеність клієнтів, то це означає, що результатом буде «мережа ланцюгів постачання» етапи різних ланцюгів співпрацюють з один одного, результатом буде «мережа ланцюгів постачання» (SCN, supply chain net).

Однією з головних цілей як GSCM, так і GSN є оцінка впливу на навколишнє середовище виробництва та/або розподілу для зменшення викидів парникових газів через логістичну діяльність [17]. Останні публікації, пов'язані з дослідженням обсягів викидів парникових газів в ланцюгах/мережах постачання, привертають увагу вчених до обґрунтування стратегічних, тактичних та операційних рішень, що стосуються різних учасників та стейкхолдерів, і доводять необхідність розроблення стратегій декарбонізації, що передбачає відстеження екологічного сліду.

Вимірювання вуглецевого сліду є важливим інструментом для підприємств, які прагнуть зменшити свій вплив на довкілля та впровадити зелені практики. Це дає можливість оцінити ефективність програм зеленого управління та зосередити зусилля на етапах виробництва та транспортування, де можна досягти більш значущого зменшення викидів парникових газів. Крім того, вимірювання вуглецевого сліду може бути корисним для споживачів, які хочуть зробити свідомий вибір та підтримати компанії, які дбають про довкілля. Деякі компанії навіть надають інформацію про вуглецевий слід своїх продуктів на упаковках або на своїх веб-сайтах, що дозволяє споживачам зробити свідомий вибір та підтримати зелені практики.

Проте, вимірювання викидів в зелених мережах постачання може бути складною задачею. Кожна ланка мережі постачання може мати свої власні викиди, які слід враховувати. Крім того, існують різні методи вимірювання викидів, що можуть використовуватися в різних етапах постачання. Наприклад, можна вимірювати викиди, пов'язані з виробництвом сировини, виготовленням продукту, транспортуванням та зберіганням.

Отже, проблема полягає в тому, що необхідно визначити правильний метод вимірювання викидів в зелених мережах постачання, щоб отримані дані були точними та надійними. Крім того, потрібно враховувати вплив зміни параметрів виробництва та транспортування на викиди, а також використовувати новітні технології та інструменти для вимірювання викидів та аналізу даних.

2.3.2 Постановка задачі оптимізації викидів парникових газів в зелених ланцюгах постачання

Зауважимо, що наразі багато організацій займаються розробкою методик вимірювання викидів в ланцюгах постачання, серед яких можна виділити:

- Міжнародна організація стандартизації (ISO) – розробляє міжнародні стандарти, включаючи стандарти для вимірювання викидів і повітряного забруднення.
- Організація Об'єднаних Націй зі збереження природних ресурсів та екосистем (UNEP) – займається оцінкою викидів і розробкою методів вимірювання викидів.
- Організація з економічного співробітництва та розвитку (OECD) – розробляє методики вимірювання викидів і забруднення в різних секторах економіки.
- Міжурядовий панель зміни клімату (IPCC) – оцінює наукову і технічну інформацію про зміну клімату, включаючи викиди парникових газів в різних секторах економіки.
- неприбуткові організації, такі як Global Reporting Initiative (GRI) та Carbon Trust – розробляють методики вимірювання викидів і забезпечують консультаційні послуги з екологічного виробництва та розвитку.
- консалтингові компанії, такі як Deloitte, PwC, EY, KPMG тощо, займаються розробкою методів вимірювання викидів і повітряного забруднення для своїх клієнтів в різних секторах економіки.

Існує кілька стандартизованих протоколів для вимірювання викидів в ланцюгах постачання, зокрема, ISO 14064 – Стандарт ISO для складання звітів про викиди парникових газів, GHG Protocol – Протокол Глобального партнерства з викидів парникових газів (Greenhouse Gas Protocol) – стандарт для обліку та звітування викидів парникових газів, PAS 2050 – Протокол PAS 2050:2008 визначає методику оцінки впливу на зміну клімату продукту або послуги від поставки сировини до кінцевого використання, Carbon Trust Standard – Стандарт Carbon Trust Standard встановлює міжнародно визнаний набір критеріїв для оцінки вуглецевого сліду організацій та продуктів тощо.

Протокол парникових газів (GHG) містить методологію вимірювання викидів в ланцюгах постачання і виділяє дві групи: викиди вгору та вниз за течією (upstream та downstream). «Upstream» означає етапи, що передують виробництву кінцевого продукту. Це можуть бути етапи добування сировини, вирощування рослин чи тварин для виробництва харчових продуктів, транспортування сировини до заводу тощо. «Downstream» означає етапи, що відбуваються після виробництва кінцевого продукту. Це можуть бути етапи збору, транспортування, зберігання та продажу кінцевого продукту. На рис. 2.3.1 графічно зображено структуру ланцюга постачання, що формується фокусною компанією.

Виділення різних етапів ланцюга постачання важливе для вибору методів вимірювання викидів в залежності від характеристик викидів та можливостей збору та аналізу даних. На етапі upstream, коли мова йде про викиди в діяльності постачальників сировини, можуть використовуватися методи, які дозволяють визначити екологічний слід виробництва цієї сировини. Ці методи можуть передбачати аналіз життєвого циклу сировини, від початкового етапу видобутку або вирощування до транспортування до наступного етапу виробництва. Зазвичай на цьому етапі важливо зібрати інформацію про кількість викидів окремих забруднюючих речовин, що утворюються під час виробництва сировини, та визначити їх ефективність управління цими викидами. На етапі downstream, коли мова йде про викиди від

постачальників кінцевих продуктів, можуть використовуватися методи, які дозволяють визначити викиди від окремих компонентів кінцевого продукту під час його використання та утилізації. Для цього можуть застосовуватися методи аналізу емісій, які дозволяють визначити кількість викидів з окремих джерел на етапі використання та утилізації кінцевого продукту.

Рис. 2.3.1. Структура ланцюга постачання та сфери прийняття рішень, що впливають на обсяг викидів

Розглянемо більш детально вимірювання викидів в ланках ланцюга постачання.

2.3.2.1 Розрахунок викидів парникових газів на виробництві.

Для розрахунку викидів парникових газів на виробництві можна використовувати різні методики, які відрізняються за точністю та складністю використання. Для малих підприємств або початкових розрахунків можна використовувати загальні методики, такі як методика «карбонового сліду», що полягає в розрахунку викидів CO_2 в еквіваленті кілограмів на одиницю продукції. Це можна зробити шляхом виміру кількості споживаної енергії та кількості викидів CO_2, які утворюються під час виробництва.

Найбільш поширеною формулою розрахунку викидів парникових газів в тонах еквіваленту CO_2 є:

$$GHG_{вир} = \sum_{i=1}^{m} F_i * EF_i * OF_i + \sum_{j=1}^{n} E_j * EF_j ,$$

де:

F_i – кількість палива i-го виду (в тонах);

EF_i – коефіцієнт викидів парникових газів для кожного виду палива;

OF_i – коефіцієнт окислення вуглецю (для вуглеводнів);

E_j – кількість спожитої електроенергії (в кВт-годинах);

EF_j – коефіцієнт викидів парникових газів для електроенергії.

Більш точні методики включають в себе розрахунок викидів кожного з парникових газів окремо та врахування впливу інших факторів, таких як рівень відходів, використання транспорту та інші. Ці методики вимагають більш детального аналізу даних та можуть бути складнішими для використання, але вони дають більш точні результати.

2.3.2.2 Розрахунок викидів парникових газів при постачанні матеріальних ресурсів

Розрахунок екологічного сліду сировини від постачальника до виробника (в тонах еквіваленту CO_2) можна розрахувати за формулою:

$$GHG_{пост} = \left(A \times B \times C \times D\right) / E,$$

де:

A – кількість виготовлених виробів / обсяг постачання;

B – викиди парникових газів при виробництві одного виробу / одиниці постачання;

C – коефіцієнт транспортування (наприклад, кілометри перевезення);

D – коефіцієнт викидів CO_2 від транспортування однієї одиниці товару на один кілометр;

E – кількість виробів/товарів, що доставляються.

Для отримання більш точних даних варто використовувати спеціалізоване програмне забезпечення для моніторингу викидів та зменшення негативного впливу на довкілля. Також важливо враховувати ефективність використання ресурсів на

2.3.2.3 Викиди парникових газів при транспортуванні

Розрахунок викидів парникових газів при транспортуванні сировини, комплектуючих, компонентів, а також готової продукції (в тонах еквіваленту CO_2) здійснюють за формулою

$$GHG_{трансп} = \left(distance \times fuel\ efficiency \times emission\ factor\right) / occupancy,$$

де:

distance – дистанція, пройдена транспортним засобом (в кілометрах);

fuel efficiency – коефіцієнт паливної ефективності транспортного засобу (в літрах / 100 км або милях на галон);

emission factor – коефіцієнт викидів парникових газів для кожного виду палива;

occupancy – кількість пасажирів в транспортному засобі.

Ця формула дає орієнтовні значення викидів, що залежать від типу транспорту та використаного виду палива.

Отже, для розрахунку викидів парникових газів на виробництві можна використовувати різні методики, які відрізняються за точністю та складністю використання. Для малих підприємств або початкових розрахунків можна використовувати загальні методики, такі як методика «карбонового сліду», що полягає в розрахунку викидів CO_2 в еквіваленті кілограмів на одиницю продукції. Це можна зробити шляхом виміру кількості споживаної енергії та кількості викидів CO_2, які утворюються під час виробництва. Існуючі методи дозволяють розрахувати обсяги викидів як в окремих ланках ланцюга постачання, так і на шляхах переміщення матеріальних ресурсів чи готового продукту. Більш точні методики включають в себе розрахунок викидів кожного з парникових газів окремо та врахування впливу інших факторів, таких як рівень відходів, використання транспорту та інші. Ці методики вимагають більш детального аналізу даних та можуть бути складнішими для використання, але вони дають більш точні результати.

Важливо розуміти, що розрахунки викидів парникових газів на виробництві є лише початковим кроком у зменшенні викидів та встановленні карбонового балансу. Для досягнення карбонної нейтральності, підприємствам необхідно розробляти різні управлінські рішення, орієнтовані на зменшення використання енергії, підвищення енергоефективності, перехід на відновлювальні джерела енергії та інші. Саме тому для оптимізації викидів в ланцюгах/мережах постачання використовують різноманітні математичні моделі, методи та алгоритми.

2.3.3 Адаптовані моделі мережевих потоків з мінімальною вартістю для мінімізації викидів парникових газів

Для мінімізації обсягів викидів парникових газів в ланцюгах/мережах постачання можуть бути використані типові задачі потокової (мережевої) оптимізації, за якими пошук оптимуму досягається мережевою моделлю у вигляді орієнтованого графу, де вузли представляють окремі ланки ланцюгів постачання (постачальників, виробників, посередників, перевізників тощо), а орієнтовані дуги визначають економічні, екологічні, інформаційні зв'язки між вузлами.

Проблема мережевого потоку з мінімальною вартістю (Minimum Cost Network Flow Problem, MCNFP) полягає у знаходженні найдешевшого способу передачі певної кількості потоку через транспортну мережу [18]. Зазвичай ця проблема формулюється як задача лінійного програмування, де треба знайти такі значення потоків на дугах мережі, які задовольняють обмеженням потоку та мінімізують вартість перенесення потоку.

У реальних задачах можуть виникати різні варіації цієї проблеми, наприклад, коли мережа має ємності та кошти перенесення можуть залежати від обсягу потоку на дугах. В таких випадках говорять про проблему мережевого потоку з мінімальною вартістю з обмеженнями (Capacitated Minimum Cost Network Flow Problem, CMCNFP) або про інші варіації цієї проблеми.

Нехай $G=(N,A)$ є направленим мережевим графом з n вузлами та m дугами, де N та A є множинами вузлів та дуг відповідно. Кожна дуга $(i,j)\in A$ має вартість c_{ij}, що у даному контексті означає обсяг викидів парникових газів в еквіваленті CO_2 на перевезення одиниці товару вздовж дуги (i,j). Кожна дуга (i,j) також пов'язана з кількістю потоку x_{ij} на дузі, нижньою межею l_{ij} та верхньою межею u_{ij} потоку; тобто $l_{ij}\le x_{ij}\le u_{ij}$. Однак, якщо $u_{ij}=+\infty$, то MCNFP називається з необмеженою пропускною здатністю (також відомим як задача з перевалками вантажу). Ми пов'язуємо з кожним вузлом $i\in N$

число b_i, яке вказує на його доступну кількість постачання або попиту. Вузол i буде називатися джерелом, стоком або проміжним вузлом, залежно від того, чи $b_i > 0$, $b_i < 0$ або $b_i = 0$ відповідно. Таким чином, можна ефективно моделювати множину реальних застосувань, що вимагають пересилання різних продуктів зі складів (вузли постачання) до ринків (вузли попиту) через деяку кількість пунктів перевезення (вузли перевалки). З урахуванням введених позначень одно продуктову задачу MCNFP можна сформулювати наступним чином:

$$\sum_{(i,j)\in A} c_{ij} x_{ij} \to \min, \tag{2.3.1}$$

$$s.t. \sum_{k:(i,k)\in A} x_{ik} - \sum_{j:(j,i)\in A} x_{ji} = b_i, \ \forall i \in N, \tag{2.3.2}$$

$$l_{ij} \le x_{ij} \le u_{ij}, \ \forall (i,j) \in A. \tag{2.3.3}$$

Обмеження (2.3.2) відображає рівняння збереження потоку, а обмеження (2.3.3) називають обмеженням на пропускну здатність дуги (ємність потоку).

Задача (2.3.1) – (2.3.3) є задачею лінійного програмування великої розмірності. Практичне використання цієї моделі передбачає декілька особливих випадків, які дозволяють спростити постановку задачі і отримати наступні моделі [18].

2.3.3.1 Транспортна задача у мережевій постановці

Транспортна задача (transshipment problem, TP) може бути представлена графом $G(S,D,A) = G(N,A)$, де S, D – дві роз'єднані множини вузлів такі, що $|S| = nS$, $|D| = nD$ і $N = S \cup D$. Позначення $|S|$ та $|D|$ становлять кількість елементів множин S та D відповідно. Вузли постачання та вузли споживання (попиту) позначені як $i \in S$ та $j \in D$ відповідно. Дуга $(i,j) \in A$ спрямована від вузлів S до вузлів D. Математична формулювання транспортної задачі з матрицею викидів розміром $nS \times nD$, вектором постачання bS розміром $nS \times 1$ та

вектором попиту bD розміром $nD \times 1$ таким, що $S \le D$, можна представити наступним чином:

$$\sum_{(i,j)\in A} c_{ij} x_{ij} \to \min, \tag{2.3.4}$$

$$s.t. \sum_{(i,j)\in A} x_{ij} = b_{S_i}, \ \forall i \in S, \tag{2.3.5}$$

$$\sum_{(i,j)\in A} x_{ij} = b_{D_j}, \ \forall j \in D, \tag{2.3.6}$$

$$x_{ij} \ge 0, \forall i \in S, \forall j \in D. \tag{2.3.7}$$

В задачі ТР здійснюються перевезення продуктів між вузлами мережі. Однак, вона також може бути ускладненою, якщо доповнити обмеженнями на місткість транспортних засобів або на доступність різних маршрутів перевезень між вузлами.

2.3.3.2 Задача призначення у мережевій постановці

Для оптимізації викидів в мережі можна використати модель задачі призначення з обмеженнями (Linear Sum Assignment Problem, LSAP англ. Linear Assignment Problem, LAP). Це класична задача комбінаторної оптимізації, яка полягає в призначенні N робітників до N завдань, коли кожен робітник може бути призначений до декількох завдань, а кожне завдання може бути призначене для декількох робітників, з обмеженням, що сума призначень для кожного робітника та кожного завдання має дорівнювати 1. У даному випадку замість матриці вартостей використовується матриця викидів таким чином, щоб знайдені призначення точок були мінімальними з точки зору сумарної вартості. Оскільки більші викиди відповідають меншим вартостям, це означає, що точки з найбільшими викидами повинні бути співставлені з тими точками, які мають найменшу вартість, і навпаки.

Математична модель LSAP полягає у знаходженні такого призначення, що мінімізує вартість призначень:

$$\sum_{(i,j)\in A} c_{ij} x_{ij} \to \min, \tag{2.3.8}$$

$$s.t. \sum_{(i,j)\in A} x_{ij} = 1, \ \forall i \in S, \tag{2.3.9}$$

$$\sum_{(i,j)\in A} x_{ij} = 1, \ \forall j \in D, \tag{2.3.10}$$

$$x_{ij} \in \{0,1\}, \ \forall i \in S, \ \forall j \in D. \tag{2.3.11}$$

Задача (2.3.8) – (2.3.11) є задачею булевого лінійного програмування і може бути розв'язана стандартними програмними продуктами.

2.3.3.3 Задача про пошук найкоротшої відстані

Задача про пошук найкоротшого шляху (Shortest Path Problem (SPP)) – це задача про знаходження найкоротшого шляху між двома вузлами в орієнтованому або неорієнтованому графі зі зваженими ребрами. У нашому випадку вага ребер може представляти обсяг викидів.

Математична модель задачі SPP полягає в тому, щоб знайти найкоротший шлях між двома заданими вузлами графу, де вага кожного ребра задана деякою невід'ємною величиною:

$$\sum_{(i,j)\in A} c_{ij} x_{ij} \to \min, \tag{2.3.12}$$

$$s.t. \sum_{k:(i,k)\in A} x_{ik} - \sum_{j:(j,i)\in A} x_{ji} = \begin{cases} 1, & \text{якщо } i = s \\ -1, & \text{якщо } i = t \\ 0, & \text{інакше} \end{cases}, \tag{2.3.13}$$

$$x_{ij} \in \{0,1\}, \ \forall (i,j) \in A. \tag{2.3.14}$$

Отже, представлені задачі мережевих потоків з мінімальною вартістю мають комбінаторну властивість, тому їх розв'язок може бути знайдений шляхом вирішення відповідної задачі лінійного програмування, якщо *b* та *u* мають цілочислові значення.

Незважаючи на те, що MCNFP та її спеціальні випадки (TP, LSAP, SPP) можуть бути ефективно вирішені за поліноміальний час, деякі узагальнення MCNFP є нерозв'язними та не можуть бути ефективно вирішені. Зокрема, у динамічній версії MCNFP вартість/викиди та місткість дуги можуть змінюватися з часом [19]. Також у вузлах мережі можуть бути обмеження на тривалість очікування/обсяг викидів у вузлі. Також NP-повною буде задача мінімально вартісного багатопродуктового потоку [20].

Автори [18] виділяють окремо задачу мінімального потоку з опуклими витратами на перевезення. У звичайній задачі мережевого потоку вартість перевезення по кожній дуги є лінійною функцією кількості перевезених одиниць товару, а в задачі з опуклими витратами ця функція може бути опуклою, тобто вартість залежить від кількості перевезених одиниць товару нелінійно. Така задача може виникати у різних областях, де витрати на перевезення залежать від пропускної здатності системи і є опуклими функціями. Наприклад, у міських транспортних системах, де вартість перевезення залежить від кількості пасажирів та стану доріг; або в мережах зв'язку, де витрати залежать від пропускної здатності мережі та рівня завантаження. Якщо витрати є кусково-опуклими функціями, то задачу можна перетворити у класичну задачу мережевого потоку з мінімальною вартістю, але це може призвести до значного збільшення кількості дуг в мережі.

Для розв’язання лінійних задач мінімально вартісних потоків часто використовують покращену версію алгоритму симплексного методу Simplex Pivot5, який має вбудовану підтримку паралельної обробки даних, що дозволяє використовувати більш потужні обчислювальні ресурси для збільшення швидкості розв'язування задач.

Для розв’язування задач мережевого потоку існують чимало обчислювальних алгоритмів. Першим поліноміальним алгоритмом для розв'язання MCNFP був алгоритм Edmonds & Karp у 1972 році [22], і з того часу було запропоновано багато варіантів алгоритмів, які використовують методи масштабування [23]. Вони дозволяють знайти розв'язок задачі в прийнятний для практичного застосування час. Знайти оптимальні рішення за мінімальну кількість ітерацій

дозволяють алгоритми Network Primal Exterior Point Simplex Algorithm (NEPSA) та Dual Network Exterior Point Simplex Algorithm (DNEPSA), які знаходять оптимальний шлях/потік, який є основним, але не завжди можливим, а також інший двоїстий, який є можливим, але не завжди основним [24].

В табл. 2.3.1 узагальнено дані про найбільш популярні солвери, які можна використовувати для мінімізації викидів в мережах постачання. В залежності від розміру і складності мережі та задачі, один солвер може бути ефективнішим за інший. Тому вибір конкретного солвера залежить від конкретних умов задачі та ресурсів, які можуть бути використані для її вирішення.

Таблиця 2.3.1. Порівняльний аналіз солверів для розв'язання задач

Назва солвера	Тип солвера	Максимальний розмір задачі	Підтримка мережевих задач	Підтримка інших типів задач
MCF	Сітковий	До 1 млн змінних	Так	Ні
RELAX-IV	Лінійного програмування	До 10 тис змінних	Ні	Так, лінійне програмування
CS2	Мережевий	До 100 тис вершин	Так	Ні
Simplex Pivot5	Лінійного програмування	До 10 тис змінних	Ні	Так, лінійне програмування
DNEPSA	Лінійного програмування	До 50 тис змінних	Ні	Так, лінійне програмування

Отже, запропоновані математичні моделі, методи та алгоритми розв'язання задач у мережевій постановці дозволять оптимізувати конфігурацію мереж постачання з точки зору мінімізації викидів парникових газів. Це важлива задача для зменшення впливу

господарської діяльності на навколишнє середовище та забезпечення сталого розвитку. За наявності достатньої кількості даних та аналітичних інструментів прийняття рішень вирішення даної проблеми може бути автоматизовано за допомогою алгоритмів машинного навчання та інших методів штучного інтелекту. Наприклад, може бути розроблена система, яка буде автоматично розраховувати оптимальний ланцюг постачання та викиди на основі зібраних даних та заданих критеріїв оптимізації. Однак, прийняття остаточного рішення повинно здійснюватися людиною, яка зможе врахувати контекст та інші недосяжні для машинного навчання фактори.

2.3.4 Метод DEA для оцінювання ефективності управлінських рішень в зелених ланцюгах постачання з орієнтацією на еталонні значення (бенчмарки)

Однією з ключових проблем управління зеленими ланцюгами постачання є оцінювання сталості постачальників з дотриманням принципів сталого розвитку. Для цього використовують різноманітні критерії сталості та продуктивності мережевих структур, а також беруть до уваги багато факторів, що впливають на результативність/ефективність прийнятих рішень. Сказане обумовлює науковий інтерес до багатокритеріальних систем прийняття рішень, і, зокрема, методу DEA.

Метод (Data Envelopment Analysis) є одним із найбільш поширених підходів до оцінювання однорідних об’єктів чи структурних організаційних одиниць (DMU, Decizion-making units) за показниками входів і виходів у відповідному вимірі [25]. Класичний DEA розглядає DMU як чорну скриньку, а оцінка ефективності/продуктивності DMU пов'язана лише зі вхідними та вихідними параметрами. Однак, у більшості випадків DMU мають мережеву структуру, наприклад, вихід першого етапу стає входом для наступного.

Ідея методу полягає у розділенні сукупності об'єктів за певними критеріями на дві групи – невелику кількість лідерів та усіх інших аутсайдерів. Відношення між використаними входами та досягнутими виходами визначають два класи оптимізаційних задач:

- вихід-орієнтований підхід до максимізації зваженої кількості виходів окремого об'єкта, який може бути отриманий за заданою зваженою кількістю входів (output-oriented model),
- вхід-орієнтований підхід до мінімізації зваженої кількості входів окремого об'єкта, щоб отримати задану зважено у кількість виходів (input -oriented model).

Зокрема, за вихід-орієнтованим підходом вирішується оптимізаційна така задача:

знайти такі вагові коефіцієнти входів $V\left(v_i,\, i=\overline{1,m}\right)$ та виходів $U\left(u_r,\, r=\overline{1,s}\right)$ для об'єкта $p\left(p=\overline{1,n}\right)$, щоб максимізувати його ефективність

$$e_p=\frac{UY_p}{VX_p}=\frac{u_1y_{p,1}+\ldots+u_ry_{p,r}+\ldots+u_zy_{p,z}}{v_1x_{p,1}+\ldots+v_ix_{p,i}+\ldots+v_mx_{p,m}},$$

за умови, що відносні оцінки ефективності усіх обєктів не перевищують 100 %:

$$\frac{UY_p}{VX_p}\le 1,\ \ j=\overline{1,n},\ V\ge 0,\ U\ge 0\,.$$

Отримана ефективність часто масштабується у діапазоні від 0 до 1, де значення ближче до 1 вказує на вищу ефективність, а значення ближче до 0 – на меншу ефективність.

Оскільки цільова функція цієї моделі та система обмежень є дробами, чисельник і знаменник яких є лінійними комбінаціями шуканих змінних, то ця задача є задачею дробово-лінійного програмування, для розвязання якої існують різноманітні методи та пакети прикладних програм. На практиці такі задачі часто зводяться до задач лінійного програмування та їх двоїстих оцінок, що зменшує кількість обмежень. Відповвідно до двоїстого підходу відбувається

мінімізація зваженої суми вхідних параметрів по відношенню до одного з нормованих вихідних параметрів.

В табл. 2.3.2 узагальнено найбільш поширені моделі DEA, які використовуються в різних галузях економіки (на основі [26]).

Табл. 2.3.2. Класифікація базових моделей методу DEA

Пряма вихід-орієнтована модель (CCRP-Output)	Двоїста вихід-орієнтована модель (CCRP-Output)
$\phi_0 \to \max$, $-\sum_{m=1}^{n} y_{jm}\lambda_m + y_{j0}\phi_0 \le 0$, $\sum_{m=1}^{n} x_{im}\lambda_m \le x_{i0}$, $\lambda_j \ge 0,\ m = \overline{1,n}$, *де* φ – величина ефективності досліджуваного підприємства, λ – фактор зважування (змінна)	$g_0 = \sum_{i=1}^{r} t_i x_{i0} \to \min$, $-\sum_{j=1}^{z} \mu_j y_{jm} + \sum_{i=1}^{r} t_i x_{im} \ge 0$, $\sum_{j=1}^{s} \mu_j y_{j0} = 1$, $\lambda_j \ge 0,\ t_i \ge 0$, де g_0 – величина ефективності досліджуваного підприємства, μ_j, t_i – фактори зважування (змінні)
Пряма вхід-орієнтована модель (CCRP-Input)	Двоїста вхід-орієнтована модель (CCRP-Input)
$\theta_0, \lambda_j \to \min$, $\sum_{m=1}^{n} y_{jm}\lambda_m \ge y_{j0},\ \forall j \in \overline{1,n}$, $x_{i0}\theta_0 - \sum_{m=1}^{n} x_{im}\lambda_m \ge 0,\ \forall i = \overline{1,n}$, $\lambda_m,\ x_i \ge 0$.	$\sum_{j=1}^{s} \mu_j y_{j0} \to \max$, $\sum_{i=1}^{r} t_i x_{i0} = 1$, $\sum_{j=1}^{s} \mu_j y_{jm} - \sum_{i=1}^{r} t_i x_{im} \le 0,\ \forall m = \overline{1,n}$,

де λ – фактор зважування (змінна)	$\mu_j, t_i \geq 0$, де μ, t – фактори зважування (змінні)

Прямі вихід- та вхід-орієнтовані моделі можуть містити обмеження, які неможливо виконати шляхом пропорційного збільшення вихідних параметрів чи зменшення вхідних факторів. Тому в роботі [27] запропоновано неархімедову константу ε, яка приймає дуже малі значення. Тоді цільова функція та обмеження для вхідних і вихідних параметрів доповнюються змінними резерву $sj+$, $si-$ (табл. 2.3.3).

Табл. 2.3.3. Моделі вихід- та вхід-орієнтованої задачі з урахуванням змінних резерву

Пряма output-орієнтована модель (CCRp-Output) зі змінною резерву	Пряма input-орієнтована модель (CCRp-Input) зі змінною резерву
$\phi_0 + \varepsilon\left(\sum_{j=1}^{s} s_j^+ + \sum_{i=1}^{r} s_i^-\right) \to \max$,	$\theta_0, \lambda_j - \varepsilon\left(\sum_{j=1}^{s} s_j^+ + \sum_{i=1}^{r} s_i^-\right) \to \min$,
$y_{j0}\phi_0 - \sum_{m=1}^{n} y_{jm}\lambda_m + s_r^+ = 0$,	$\sum_{m=1}^{n} y_{jm}\lambda_m - s_j^+ = y_{j0}$,
для всіх фірм $j = \overline{1,s}$,	для всіх фірм $j = \overline{1,s}$,
$\sum_{m=1}^{n} x_{im}\lambda_m + s_i^- = x_{i0},\ \forall m = \overline{1,n}$,	$x_{i0}\theta_0 - \sum_{m=1}^{n} x_{im}\lambda_m - s_i^- = 0,\ \forall i = \overline{1,r}$,
$\lambda_m, s_j^+, s_i^- \geq 0,\ \forall m = \overline{1,n}$.	$\lambda_m, s_j^+, s_i^- \geq 0,\ \forall m = \overline{1,n}$.

Представлені в табл. 2.3.2 і табл. 2.3.3 моделі можуть бути використані для оцінки ефективності діяльності окремого підприємства як складного об'єкту з безліччю входів (витрат) та виходів (випуском продукції), а також для порівняння його діяльності

з іншими схожими компаніями у навколишньому середовищі функціонування.

Розглянемо можливості використання методу DEA для оптимізації рішень в управлінні зеленими ланцюгами постачання. В роботі [28] автори дослідили взаємозвязок між ефективністю управління в зелених ланцюгах постачання та обсягами викидів, що мають негативний вплив на довкілля. Вони перевірили значущість трьох зелених факторів, таких як рівень обслуговування, викиди (CO_2) та розмір ланцюга (дуги), на ефективність всієї системи до і після модифікації потенційних DMU та порівняли результати. Знаходження потенційних DMU серед неефективних, які мають можливість стати ефективними після модифікації згідно з бенчмарками (кращими зразками, еталонними значеннями).

Метод DEA базується на непараметричному підході, що дозволяє враховувати як позитивні (бажані) результати, так і негативні (небажані) результати в екологічній оцінці. Небажані виходи негативно впливають на навколишнє середовище (збільшення викидів CO_2, утворення відходів, бракована продукція тощо); таким чином, вони є шкідливими факторами для ланцюгів постачання. Слід зазначити, що в більшості випадків небажані результати є побічними продуктами бажаних результатів.

Розглянемо більш детально постановки задач та опис математичних моделей DEA для обґрунтування рішень в зелених ланцюгах постачання.

Нехай DMU_0 позначає одиницю із загальної кількості n одиниць, відносна ефективність яких оцінюється. Визначимо $x_0 \in Rm_+$ та $y_0 \in Rs_+$ як входи та виходи DMU_0. Нехай (x, y, v) – вектор, що складається з m входів, s бажаних і l небажаних виходів. Бажаний вектор виходу ' y ' є нульовим з'єднанням із небажаними виходами ' v ', якщо $(y, v) \in P(x)$, $v = 0$, тоді $y = 0$.

Будемо вважати $P(x)$ вихідним набором. Це означає, що якщо виробляються деякі хороші (бажані) результати, то неминуче, як

наслідок, також виробляються деякі погані (небажані) результати. Іншими словами, враховуючи нульову спільність, можна зробити висновок, що якщо вихідний вектор (y,v) є здійсненним і погані результати не виробляються, то можна отримати лише нульовий хороший результат. В роботі [29] описана ідея нульових спільних результатів і слабкої одноразової можливості випусків, а також визначено слабку та сильну розпорошуваність виходу. Слабка розпорошуваність (weak disposability) виходу передбачає, що якщо $(y,v)\in P(x)$ і $0\le\theta\le 1$, тоді $(\theta y,\theta v)\in P(x)$. Це означає, що якщо утримувати входи x постійними, то бажані виходи повинні бути зменшені за рахунок зменшення небажаних виходів. Враховуючи технологію виробництва, яка генерує як бажані, так і небажані продукти (при цьому викид небажаного виходу не є безпосередньо безнаслідковим), слабка розпорошуваність виходів стверджує, що при фіксованих вхідних параметрах x, зменшення небажаних виходів повинно супроводжуватись зменшенням бажаних виходів.

Сильна розпорошуваність (strong disposability) виходу означає, що якщо у просторі $(\bar{y},\bar{v})\in P(x)$ і $(y,v)\le(\bar{y},\bar{v})$, тоді $(y,v)\in P(x)$. Відповідно до цього визначення будь-який вихід, який несумісний з технологією, може бути викинутий з наслідками. Згідно з вищезазначеними визначеннями, сильна розпорошуваність може включати в себе слабку розпорошуваність, але не навпаки [29].

З урахуванням зроблених припущень класична двохетапна модель ДЕА має наступний вигляд:

$$\theta-\varepsilon\left(\sum_{i=1}^{m}s_i^-+\sum_{r=1}^{s}s_i^+\right)\to\min,$$

$$\sum_{j=1}^{n}\lambda_j x_{ij}+s_i^+=\theta x_{i0},\ \ \forall i=\overline{1,m},$$

$$\sum_{j=1}^{n}\lambda_j y_{rj}-s_r^+=y_{r0},\ \ \forall r=\overline{1,s},$$

$$\lambda_j\ge 0,\ \ \forall j=\overline{1,n},$$

$$s_i^- \geq 0,\ s_r^+ \geq 0,\ \forall i = \overline{1,m},\ \forall r = \overline{1,s}.$$

На першому етапі розглядається радіальне зменшення вхідних параметрів, а на другому – максимізується сума резервів вхідних параметрів та надлишків вихідних параметрів. Слід зазначити, що небажані вихідні продукти в основному є побічними продуктами бажаних вихідних продуктів. Для подальшого зменшення небажаних вихідних продуктів необхідно зменшити бажані вихідні продукти в різних ланках ланцюга постачання. Для врахування зв'язків між виробництвом бажаних і небажаних вихідних продуктів та оцінювання ефективності системи виробництва в цілому запропонована модель 2.

$$\alpha + \varepsilon\left(\sum_{i=1}^{m} s_i^- + \sum_{r=1}^{l} s_r^+ + \sum_{r_1=1}^{s_1} \bar{s}_r^{-+} + \sum_{p=1}^{1}\left(n_{po} - t_{po}\right) + \sum_{f=1}^{h} q_f^+ + \sum_{f=1}^{h} q_f^-\right) \to \max,$$

$$\sum_{j=1}^{n} \lambda_j^1 x_{ij} + s_i^- = x_{i0} - \alpha d_i^x,\ \forall i = \overline{1,m}, \tag{a}$$

$$\sum_{j=1}^{n} \lambda_j^1 z_{fj} - q_f^- = z_{f_0} + \alpha d_f^z,\ \forall f = \overline{1,h}, \tag{b}$$

$$\sum_{j=1}^{n} \lambda_j^2 z_{fj} + q_f^+ = z_{f_0} - \alpha d_f^z,\ \forall f = \overline{1,h}, \tag{c}$$

$$\sum_{j=1}^{n} \lambda_j^2 v_{pj} + n_{po} - t_{po} = v_{po}, \tag{d}$$

$$\sum_{j=1}^{n} \lambda_j^2 y_{rj} - s_r^+ = y_{ro} + \alpha d_r^y,\ r = 1, \tag{e}$$

$$\sum_{j=1}^{n} \lambda_j^2 y_{r_1 j} - \bar{s}_{r_1}^{-+} = y_{r_1 o} + \alpha d_{r_1}^y,\ \forall r_1 = \overline{1,s_1}, \tag{f}$$

$$\sum_{j=1}^{n} \lambda_j^2 v_{1j} \leq f\left(\sum_{j=1}^{n} \lambda_j^2 y_{1j}\right), \tag{g}$$

$$n_{po} t_{po} = 0,\ p = 1, \tag{h}$$

$$n_{po} \geq 0,\ t_{po} \geq 0,\ \alpha \geq 0,\ p = 1, \tag{i}$$

$$\lambda_j^1 \geq 0,\ \lambda_j^2 \geq 0, \tag{j}$$

$$s_i^- \geq 0,\ s_r^+ \geq 0,\ \forall i = \overline{1,m},\ r = 1, \tag{k}$$

$$q_f^+ \geq 0,\ q_f^- \geq 0,\ \bar{s}_{r_1}^{+} \geq 0,\ \forall f = \overline{1,h},\ \forall r_1 = \overline{1,s_1}, \tag{l}$$

В моделі 2 y є бажаним вихідним продуктом, а v – відповідно небажаним вихідним продуктом, який є побічним продуктом бажаного вихідного продукту.

Для аналізу постачальників сировини у ланцюзі постачання може бути використано модель (3), яка має такий вигляд:

$$\alpha + \varepsilon \left(\sum_{i=1}^{m} s_i^- + \sum_{f=1}^{h} q_f^+ \right) \to \max,$$

$$\sum_{j=1}^{n} \lambda_j^1 x_{ij} + s_i^- = x_{i0} - \alpha d_i^x,\ \forall i = \overline{1,m}, \tag{a}$$

$$\sum_{j=1}^{n} \lambda_j^1 z_{fj} - q_f^+ = z_{f0} + \alpha d_i^z,\ \forall f = \overline{1,h}, \tag{b}$$

$$\alpha \geq 0,\ \lambda^1 \geq 0,\ s^- \geq 0,\ q^+ \geq 0.$$

Для аналізу виробника та боротьби з небажаним виходом цього етапу ланцюга постачання запропоновано модель 4, яка виглядає таким чином:

$$\alpha + \varepsilon \left(\sum_{r=1}^{1} s_r^+ + \sum_{r_1=1}^{s_1} \bar{s}_r^{+} + \sum_{p=1}^{1} \left(n_{po} - t_{po} \right) + \sum_{f=1}^{h} q_f^- \right) \to \max,$$

$$\sum_{j=1}^{n} \lambda_j^2 z_{fj} + q_f^- = z_{f0} - \alpha d_f^z,\ \forall f = \overline{1,h}, \tag{a}$$

$$\sum_{j=1}^{n} \lambda_j^2 v_{pj} + n_{po} - t_{po} = v_{po},\ p = 1, \tag{b}$$

$$\sum_{j=1}^{n}\lambda_j^2 y_{rj} - s_r^+ = y_{r_0} + \alpha d_r^y,\ r=1, \tag{c}$$

$$\sum_{j=1}^{n}\lambda_j^2 y_{r_1 j} - \overline{-s_{r_1}^+} = y_{r_1 o} + \alpha d_{r_1}^y,\ \forall r_1 = \overline{1, s_1}, \tag{d}$$

$$\sum_{j=1}^{n}\lambda_j^2 v_{1j} \le f\left(\sum_{j=1}^{n}\lambda_j^2 y_{1j}\right), \tag{e}$$

$$n_{po} t_{po} = 0,\ p=1, \tag{f}$$

$$n_{po} \ge 0,\ t_{po} \ge 0,\ \alpha \ge 0,\ p=1, \tag{g}$$

$$\lambda_j^2 \ge 0,\ \forall j = \overline{1, n}, \tag{h}$$

$$s_r^+ \ge 0,\ \forall i = \overline{1, m},\ r=1, \tag{i}$$

$$q_f^- \ge 0,\ \overline{-s_{r_1}^+} \ge 0,\ \forall f = \overline{1, h},\ \forall r_1 = \overline{1, s_1}, \tag{j}$$

Таким чином, запропоновані моделі 1–4 відображають процес руху матеріальних потоків від постачальників сировини через виробничі та посередницькі структури до кінцевого споживача. Метою кожного етапу цього руху є мінімізація споживання ресурсів і максимізація виробництва бажаних результатів. Оскільки виробництво готової продукції та її доставляння до споживача супроводжується небажаними факторами, зокрема, викидами парникових газів, утворенням відходів виробництва та зберігання товарів, поверненнями бракованої продукції тощо, то найкращою стратегією боротьби з ними є розгляд спільного виробництва бажаних і небажаних результатів. Наявність таких виходів відіграє важливу роль, як фактори середовища, в оцінці ефективності DMU.

Запропонована сукупність моделей DEA як процедура математичної оптимізації дозволяє зменшити вплив небажаних факторів та максимізувати ефективність/продуктивність ланцюгів постачання в цілому. Це означає, що використання об'єднаних ресурсів учасників ланцюга постачання найефективнішим способом допомагає надавати конкурентоспроможні та економічно ефективні

продукти та послуги. Запропонований багатоетапний підхід DEA допомагає оцінити ефективність сталого ланцюга постачання, коли існує довільна кількість постачальників, виробників, посередників, дистриб'юторів і споживачів, оскільки кожен етап має неоднакову вагу. Крім того, запропонована модель може дозволити особам, які приймають рішення, одночасно мінімізувати шкідливий вплив на навколишнє середовище та максимізувати експлуатаційні показники, враховуючи задоволеність клієнтів.

2.3.5 Висновки

Не дивлячись на те, що екологічні аспекти управління ланцюгами постачання є об'єктом дослідження багатьох вчених з різних країн, проблема зменшення викидів парникових газів, які є ключовими факторами глобального потепління, є недостатньо дослідженою і потребує розробки спеціальних методів вимірювання екологічного сліду в ланцюгах/мережах постачання. Впровадження бізнес-моделей з низьким рівнем викидів вуглецю може каталізувати розвиток інноваційних підходів до оптимізації логістичних мереж, розташування місць виробництва продукції, розподільчих центрів, екодизайну товарів та їх зберігання, а також залучення інвестицій в екологічні проекти.

Проведений аналіз методичних підходів до вимірювання екологічного сліду в ланцюгах постачання дозволив зробити висновок про використання математичного апарату задач потокової (мережевої) оптимізації, в яких пошук оптимуму досягається мережевою моделлю у вигляді орієнтованого графу, де вузли представляють окремі ланки ланцюгів постачання (постачальників, виробників, посередників, перевізників тощо), а орієнтовані дуги визначають економічні, екологічні, інформаційні зв'язки між вузлами. Запропоновано адаптовані до спеціальних умов задачі лінійного програмування, де треба знайти такі значення потоків на дугах мережі, які мінімізують сумарні обсяги викидів парникових газів у товарному русі від постачальників до споживачів готової продукції.

Обгрунтовано використання методів і моделей DEA, що розглядають залежність ефективності/продуктивності ланцюгів постачання між бажаними і небажаними результатами на кожному з етапів доставки сировини від постачальників до виробників продукції, а від них – до кінцевих споживачів. Небажані фактори пов'язані з вимірюванням шкідливого впливу товарного руху на довкілля.

Список літератури

1. Friedlingstein P. and all. Global Carbon Budget. *Earth Syst. Sci. Data*, 2022. No 14. Pp. 4811–4900.
2. Supply Chain Predictions: Resiliency, Sustainability And Visibility Set New Expectations.
   URL: https://www.forbes.com/sites/sap/2022/12/15/2023-supply-chain-predictions-resiliency-sustainability-and-visibility-set-new-expectations/?sh=4c261f8b52cd
3. Srivastava S.K. Green supply-chain management: a state-of-the-art literature review. *International Journal of Management Reviews*. 2007. Vol. 9. No 1. Pp. 53–80.
4. Dekker R., Bloemhof J., Mallidis I. Operations research for green logistics – an overview of aspects, issues, contributions and challenges. *European Journal of Operational Research*. 2012.Vol. 219, No. 3. Pp. 671–679.
5. Fahimnia B., Sarkis J., Davarzani H. Green supply chain management: a review and bibliometric analysis. *International Journal of Production Economics*. 2015. Vol. 162. Pp. 101–114.
6. Rajeev A., Pati R.K., Padhi S.S., Govindan K. Evolution of sustainability in supply chain management: A literature review. *J. Clean. Prod.* 2017. No 162. Pp. 299–314.
7. Diabat A., Simchi-Levi, D. A carbon-capped supply chain network problem. IEEE International Conference on Industrial Engineering and Engineering Management. Transportation Research Part D. *Transport and Environment.* 2009. Vol. 17. No 5. Pp. 370–379.

8. Bowen F.E., Cousins P.D., Lamming R.C., Farukt A.C. The role of supply management capabilities in green supply. *Prod. Oper. Manag*. 2001. 10. Pp. 174–189.
9. Asha L.N., Dey A., Yodo N., Aragon, L.G. Optimization Approaches for Multiple Conflicting Objectives in Sustainable Green Supply Chain Management. *Sustainability* 2022, 14, 12790.
10. Rajeev A., Pati R.K., Padhi S.S., Govindan K. Evolution of sustainability in supply chain management: A literature review. *J. Clean. Prod*. 2017. No 162. Pp. 299–314.
11. Sarkis J., Zhu Q.H., Lai K.H. An organizational theoretic review of green supply chain management literature. *International Journal of Production Economics*. 2011. No 130. Pp. 1–15.
12. Kumar S., Teichman S., Timpernagel T. A green supply chain is a requirement for profitability. *International Journal of Production Research.* 2012. Vol. 50. No 5. Pp. 1278–1296.
13. Judge W.Q., Elenkov D. Organizational capacity for change and environmental performance: An empirical assessment of Bulgarian firms. *J. Bus. Res*. 2005. **58**. Pp. 893–901.
14. Memari A., Rahim A.R.A., Ahmad R., Hassan A. A literature review on green supply chain modelling for optimising CO2 emission. *Int. J. Operational Research*. 2016. Vol. 26. No. 4. Pp. 509–525.
15. Corominas A. Supply chains: what they are and the new problems they raise. *International Journal of Production Research*. 2013. 51(23–24). Pp. 6828–6835.
16. Matinrada N., Roghaniana E., Rasib Z. Supply chain network optimization: A review of classification, models, solution techniques and future research. *Uncertain Supply Chain Management*. 2013. No 1. Pp. 1–24.
17. Absi N., Dauzère-Pérès S., Kedad-Sidhoum S., Penz B., Rapine C. Lot sizing with carbon emission constraints. *European Journal of Operational Research*, 2013. Vol. 227. No 1. Pp. 55–61.
18. Sifaleras A. Minimum cost network flows: problems, algorithms, and software. *Yugoslav Journal of Operations Research*. 2013. No 1. Pp. 3–17.

19. Nasrabadi E., Hashemi S.M. Minimum cost time-varying network flow problems. Optimization Methods and Software. 2010. No 25(3). Pp. 429–447.
20. Even S., Itai A., Shamir A. On the complexity of timetable and multicommodity flow. Problems. *SIAM Journal on Computing*. 1976. No 5 (4). Pp. 691–703.
21. Sifaleras A. Minimum cost network flows: Problems, algorithms, and software. Yugoslav journal of operations research. 2013. №1 (23). Pp. 3–17.
22. Edmonds J., Karp R.M.Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, 1972. No 19(2). Pp. 248–264.
23. Hamacher H.W., Pedersen C.R., Ruzika S.Multiple objective minimum cost flow problems: A review. *European Journal of Operational Research*, 2007. No 176 (3). P.1404–1422.
24. Geranis G., Paparrizos K., Sifaleras A. On a dual network exterior point simplex type algorithm and its computational behavior. *RAIRO – Operations Research*. 2012. No 46(3). Pp. 211–234.
25. Charnes A., Cooper W., Rhodes E. Measuring the efficiency of decision making units. *Eur. J. Oper. Res.* 1978. No 2. Pp. 429–444.
26. Lissitsa A., Babiéceva T. Анализ оболочки данных (DEA) – современная методика определения эффективности производства: Discussion Paper. 2003. No. 50. https://nbn-resolving.de/urn:nbn:de:gbv:3:2-23263
27. Charnes A., Cooper W.W., Rhodes E. Measuring the efficiency of decision making units. *European Journal of Operational Research*. 1978. №2. Pp. 429–444.
28. Zaare T.F., Daneshvar. S. Benchmark Approach for Efficiency Improvement in Green Supply Chain Management with DEA Models. *Sustainability*. 2023. No 15. P. 4433. DOI: 10.3390/su15054433
29. Färe R., Grosskopf S. Nonparametric Productivity Analysis with Undesirable Outputs: Comment. *American Journal of Agricultural Economics*. 2003. No 85(4). Pp. 1070–1074. DOI.10.1111/1467-8276.00510.

2.4 МАТЕМАТИЧНІ МОДЕЛІ, МЕТОДИ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ПЛАНУВАННЯ МІЖГАЛУЗЕВИХ СТРУКТУРНО-ТЕХНОЛОГІЧНИХ ЗМІН

П. І. Стецюк, О. А. Березовський, О. П. Лиховид

Анотація. Обговорюється питання аналізу та планування міжгалузевих структурно-технологічних змін за допомогою трьох оптимізаційних задач з використанням таблиць «витрати-випуск». Перша задача передбачає максимізацію сукупних доходів споживачів або/та мультиплікатора «приріст доходів – приріст виробництва» шляхом зміни структури витрат та доданої вартості за умови не інфляційного зростання доходів. Друга та третя задачі відповідають тим же критеріям, що і перша задача, але вони пов’язані зі зміною індексів цін та врахуванням характеристик наявних технологій та ресурсів. Для першої та другої оптимізаційних задач описано алгоритми їх розв’язання на основі методів негладкої оптимізації та відповідне програмне забезпечення.

Abstract. The issue of analysis and planning of cross-industry structural and technological changes is discussed with help of three optimization problems using "cost-output" tables. The first task provides maximization of the total income of consumers or/and the multiplier "income increase – production increase" by changing the structure of costs and added value under the condition of non-inflationary growth of incomes. The second and the third tasks meet the same criteria as the first task, but they are related to the change of price indices and taking into account the characteristics of available technologies and resources. For the first and the second optimization problems algorithms based on non-smooth optimization methods and corresponding software are described.

2.4.1 Вступ

Виявлення структурно-технологічних диспропорцій, які впливають на кризові явища в економіці, та аналіз шляхів їх усунення потребують широкого застосування кількісних методів досліджень, зокрема, математичного моделювання. Однією з найважливіших цілей структурно-технологічних змін є скорочення витрат виробництва. Цього можна досягти подвійним шляхом: по-перше, змінюючи технології, що використовуються, з тим, щоб зменшити питомі витрати сировини, матеріалів та інших видів продукції виробничого призначення, по-друге, скороченням застосування окремих, найбільш енерго- та ресурсномістких технологій, аж до відмови від них. Обидва ці шляхи пов'язані зі змінами на міжгалузевому рівні, що й обумовлює вибір міжгалузевих агрегованих балансових моделей як інструментарію для аналізу та планування структурно-технологічних змін.

Таблиці «витрат-виходу» Леонтьєва виявилися достатньо зручним інструментом для аналізу економічних питань. У моделях типу Леонтьєва матриця технологічних коефіцієнтів (матриця прямих витрат) передбачається відомою і розраховується на основі статистичної інформації з таблиць «витрат-вихід». М.В. Михалевич сформулював обернену задачу: як визначити ті структурно-технологічні зміни, які б знизили собівартість продукції і, тим самим, підвищили б доходи кінцевих споживачів та зробили економіку динамічнішою. Або, іншими словами, як підібрати або налаштувати технічні коефіцієнти для покращення властивостей економічного процесу. Ці моделі М.В. Михалевича [1–3] можна назвати оберненими моделями типу Леонтьєва. Нижче приведено ряд математичних моделей такого типу. Задача оптимізації планування структурно-технологічних змін може бути сформульована як задача максимізації сукупних доходів споживачів або як задача максимізації мультиплікатора "приріст доходів–приріст виробництва" за рахунок зміни структури витрат та доданої вартості за умови неінфляційного зростання доходів. У загальному випадку ці задачі є

багатоекстремальними. Для кожної з них розглянуто дві можливі постановки у формі задач математичного програмування.

Перша постановка пов'язана зі знаходженням глобального максимуму цільової функції складного виду при опуклих обмеженнях. Для неї розроблено метод знаходження локальних екстремумів на основі r-алгоритму (програма MULSTR) і запропоновано схему метода для знаходження ефективних локальних екстремумів на основі субградієнтних алгоритмів недиференційовної оптимізації у поєднанні з елементами РЕСТАРТ-технологій. Друга постановка пов'язана з можливістю зведення обох задач до моделей з простішими функціями (за допомогою введення нових змінних), для якої запропоновано аналогічний метод знаходження локальних екстремумів (програма MULSTR1).

Слідом за цими задачами (максимізація сукупних доходів споживачів або мультиплікатора "приріст доходів–приріст виробництва" за рахунок зміни структури витрат та доданої вартості) розглянуто моделі, які є їх подальшим розвитком. У першій з них керованими параметрами, що впливають на коефіцієнти прямих витрат, є ціни на продукцію галузей (на відміну від попередніх моделей, які пов’язані зі знаходженням змін до існуючих значень компонент технологічної матриці). При цьому технологічна матриця визначається за правилом $A = PA_0P^{-1}$, де P – діагональна матриця, компонентами якої є індекси цін. Для розв'язування отриманої багатоекстремальної задачі застосовано той же підхід, що і для попередніх задач (програма MULSTR2).

У Додатках В та Г наведено коди підпрограм MULSTR1 і MULSTR2 на мові RATFOR (препроцесор Фортрану) відповідно.

2.4.2 Базові відомості

В цьому підрозділі введемо основні припущення та позначення, необхідні для подальшого викладення матеріалу.

Нехай економіка країни сформована n агрегованими галузями і

$A = \{a_{ij}\}$ – матриця коефіцієнтів прямих витрат для цих галузей. Через y_i і x_i позначимо кінцевий і валовий продукт i-ї галузі у фіксованих цінах; ці величини пов'язані співвідношенням

$$x_i = \sum_{j=1}^{n} a_{ij} x_j + y_i, \quad i = \overline{1,n}. \tag{2.4.1}$$

Нехай $x = (x_1, \ldots, x_n)$ і $y = (y_1, \ldots, y_n)$, E – одинична матриця розміру $n \times n$. Співвідношення (2.4.1) означає, що вектори x (валовий продукт) і y (кінцевий продукт) пов'язані такими співвідношеннями у матричній формі:

$$y = (E - A)x \quad \text{або} \quad x = (E - A)^{-1} y. \tag{2.4.2}$$

Припустимо лінійну залежність оплати праці від обсягів виробництва в галузях. Нехай q_i – частка заробітної платні та інших виплат за працю у ціні продукції i галузі; $q = (q_1\, q_2 \ldots q_n)^T$. Загальні доходи споживачів D дорівнюють

$$D = \sum_{i=1}^{n} q_i x_i = (q, x). \tag{2.4.3}$$

Будемо вважати, що кінцевий продукт галузей складається з двох частин, одна з яких залежить, а інша – не залежить від D. Припустивши лінійну залежність першої з них від величини доходів споживачів, отримаємо

$$y_i = \alpha_i D + h_i, \quad i = \overline{1,n}, \tag{2.4.4}$$

де коефіцієнти α_i відображають, в основному, структуру індивідуального споживання та внутрішніх інвестицій, а h_i визначаються експортно-імпортним сальдо галузей та потребами колективного (у тому числі суспільного) споживання.

Виразимо D через A і q. Використовуючи (2.4.2) маємо $D = (q, x) = (q, (E - A)^{-1} y)$, звідки з урахуванням (2.4.4) отримуємо

$$D = \frac{q^T (E - A)^{-1} h}{1 - q^T (E - A)^{-1} \alpha}, \qquad (2.4.5)$$

де $h = (h_1, \ldots, h_n)$, $\alpha = (\alpha_1, \ldots, \alpha_n)$.

Визначимо ще одну функцію від A і q – мультиплікатор «приріст доходів–приріст виробництва»

$$k = q^T (E - A)^{-1} \alpha. \qquad (2.4.6)$$

Завдання полягає у визначенні таких змін елементів матриці A і вектора $q = \{q_1, \ldots, q_n\}$, які максимізували б величину D кінцевих доходів споживачів (або ж мультиплікатор «приріст доходів–приріст виробництва» k) без додаткових інфляційних впливів.

Згідно [4] умови, що виключають розвиток інфляції витрат під впливом внутрішніх факторів, мають вигляд:

$$\sum_{i=1, i \neq j}^{n} \frac{a_{ij}}{1 - a_{jj} - \bar{q}_j} < 1, \quad j = \overline{1, n}, \qquad (2.4.7)$$

де $\bar{q}_j$ – частка доданої вартості в ціні продукції j-ої галузі.

Відзначимо також, що величину $\bar{q}_j$ можна представити таким чином:

$$\bar{q}_j = l_j q_j + d_j,$$

де l_j – мультиплікатор витрат на оплату праці в j-ій галузі, а d_j – частка інших складових доданої вартості в ціні продукції j-ої галузі.

З огляду на деяку неточність вхідних даних і необхідність наявності резерву для безінфляційного збільшення компонентів доданої вартості, що не залежать від q, доцільно замість (2.4.7) розглядати умови

$$\sum_{i=1, i \neq j}^{n} \frac{a_{ij}}{1 - a_{jj} - (l_j q_j + d_j)} \leq \beta, \quad i = \overline{1, n}, \qquad (2.4.8)$$

де $\beta < 1$ – заздалегідь задана гранична величина.

2.4.3 Постановка задачі

Нехай задані технологічна матриця A, вектор q і відповідні вектори α, h, l і q. Нехай Δa_{ij} і Δq_j ($i=\overline{1,n}, j=\overline{1,n}$) – зміни існуючих значень компонент матриці A і вектора q; $\Delta q=(\Delta q_1,\ldots,\Delta q_n)$ і $\Delta A=\{\Delta a_{ij}\}_{ij}^{nn}$.

Опишемо змістовну постановку задачі визначення таких значень Δq і ΔA, що максимізували б величину D (або мультиплікатор «приріст доходів–приріст виробництва» k) без додаткових інфляційних впливів, що задається за допомогою співвідношень (2.4.8).

Цільовими функціями задачі будуть такі:

$$f_1(\Delta A,\Delta q)=\frac{(q+\Delta q)^T(E-(A+\Delta A))^{-1}h}{1-(q+\Delta q)^T(E-(A+\Delta A))^{-1}\alpha} \qquad (2.4.9)$$

відповідає величині D згідно (4.5), а

$$f_2(\Delta A,\Delta q)=(q+\Delta q)^T(E-(A+\Delta A))^{-1}\alpha \qquad (2.4.10)$$

відповідає мультиплікаторові «приріст доходів – приріст виробництва» k згідно (2.4.6).

Основні обмеження для Δq і ΔA, що дозволяють виключити додаткові інфляційні впливи, описуються за допомогою співвідношення (2.4.8). Вони мають таку форму

$$\sum_{i=1,i\neq j}^{n}\frac{a_{ij}+\Delta a_{ij}}{1-(a_{jj}+\Delta a_{jj})-(l_j(q_j+\Delta q_j)+d_j)}\le\beta,\quad j=\overline{1,n}, \qquad (2.4.11)$$

Щоб дотримувався фізичний зміст коефіцієнтів нової матриці і нового вектора, розглянемо групу обмежень

$$0\le q_j+\Delta q_j\le 1,\quad 0\le a_{ij}+\Delta a_{ij}\le 1,\quad i,j=\overline{1,n}, \qquad (2.4.12)$$

а щоб дотримувався фізичний зміст обмежень (2.4.11) розглянемо групу обмежень

$$a_{jj}+\Delta a_{jj}+l_j(q_j+\Delta q_j)+d_j\le 1,\quad j=\overline{1,n}, \qquad (2.4.13)$$

яка відповідає додатності знаменників в обмеженнях (2.4.11).

До останніх можуть бути додані обмеження на можливі границі змін коефіцієнтів прямих витрат, обумовлені особливостями існуючих технологій:

$$\underline{\gamma_{ij}} \le \Delta a_{ij} \le \overline{\gamma_{ij}} \quad i, j = \overline{1,n}, \tag{2.4.14}$$

$$\underline{\gamma_{i}} \le \Delta q_{i} \le \overline{\gamma_{i}} \quad i = \overline{1,n}, \tag{2.4.15}$$

За допомогою обмежень (2.4.14) та (2.4.15) можна керувати кількістю коефіцієнтів у технологічній матриці або векторі часток заробітної платні, які дозволяється змінювати. Для цього достатньо інші компоненти зафіксувати на конкретних значеннях, встановивши для них нижні і верхні границі рівними цим значенням.

Частина обмежень обумовлена ресурсами, що виділяються для структурно-технологічних змін:

$$\sum_{j=1}^{n}\sum_{i=1}^{n} b_{kij} \max(0, -\Delta a_{ij}) \le B_k, \quad k = \overline{1,K}, \tag{2.4.16}$$

де K – кількість ресурсів, B_k – обсяг k-го ресурсу, який виділяється з метою зниження витрат виробництва, b_{kij} – витрата цього ресурсу при реалізації заходів, що забезпечують одиничне зменшення питомих витрат продукції i-ої галузі на виробництво одиниці продукції j-ої галузі.

Надалі будемо розглядати дві постановки задачі:

1. Задача максимізації величини D, тобто знайти такі значення Δq і ΔA, які б максимізували значення функції (2.4.9) при обмеженнях (2.4.11) – (2.4.16).
2. Задача максимізації величини k (мультиплікатора «приріст доходів–приріст виробництва»), тобто знайти такі значення Δq і ΔA, які б максимізували значення функції (2.4.10) при обмеженнях (2.4.11) – (2.4.16).

2.4.4 Оптимізаційні задачі та їхній аналіз

Приведемо до більш зручного вигляду систему обмежень (2.4.11) – (2.4.16).

По-перше, позбудемося дробово-лінійних нерівностей (2.4.11). Враховуючи, що в j-ому обмеженні типу (2.4.11) знаменник доданка не залежить від i, і, крім того, з (2.4.13) випливає його додатність, групу обмежень (2.4.11) можна записати у формі лінійних нерівностей:

$$\beta(a_{jj} + \Delta a_{jj}) + \beta(l_j(q_j + \Delta q_j) + d_j) + \\ + \sum_{i=1, i\neq j}^{n} (a_{ij} + \Delta a_{ij}) \le \beta, \quad j = \overline{1,n}. \qquad (2.4.17)$$

Тоді разом з обмеженнями

$$a_{jj} + \Delta a_{jj} + l_j(q_j + \Delta q_j) + d_j \le 1, \quad j = \overline{1,n}, \qquad (2.4.18)$$

вони будуть повністю задавати вимоги, які виключають розвиток інфляції витрат під впливом внутрішніх факторів, з деяким запасом, заданим коефіцієнтом β. Однак, на відміну від обмежень (2.4.11) вони вже будуть лінійними нерівностями.

Другими розглянемо обмеження, пов'язані з ресурсами, які виділяються для структурно-технологічних змін, замінивши їхню форму на

$$\sum_{j=1}^{n}\sum_{i=1}^{n} b_{kij}^{-} \max(0, -\Delta a_{ij}) + \sum_{j=1}^{n}\sum_{i=1}^{n} b_{kij}^{+} \max(0, \Delta a_{ij}) \le B_k, \quad k = \overline{1,K}, \qquad (2.4.19)$$

де K – кількість ресурсів, B_k – обсяг k-го ресурсу, необхідного для зниження витрат виробництва, b_{kij}^{-} – витрати k-го ресурсу при реалізації заходів, що забезпечують одиничне зменшення питомих витрат продукції i-ої галузі на виробництво одиниці продукції j-ої галузі, b_{kij}^{+} – витрати k-го ресурсу при реалізації заходів, що забезпечують одиничне збільшення питомих витрат продукції i-ої галузі на виробництво одиниці продукції j-ої галузі.

Відзначимо, що з обмежень (2.4.19) випливають обмеження (2.4.16), якщо встановити коефіцієнти b_{kij}^{-} рівними коефіцієнтам b_{kij}, а коефіцієнти b_{kij}^{+} рівними нулеві. Ці обмеження в цьому розділі будемо використовувати як опуклі нерівності, що вимагає виконання умови, щоб усі коефіцієнти b_{kij}^{-} і b_{kij}^{+} були невід'ємними. Однак, варто відзначити, що за допомогою заміни змінних обмеження (2.4.19) також можна привести до форми лінійних нерівностей. Це буде супроводжуватися введенням нових змінних, кількість яких буде дорівнює подвоєній кількості невідомих коефіцієнтів матриці.

Останніми включимо обмеження, пов'язані з установкою нижніх і верхніх границь на невідомі коефіцієнти. Їх можна сформулювати в такій формі:

$$\underline{\Delta a_{ij}} \le \Delta a_{ij} \le \overline{\Delta a_{ij}}, \quad i, j = \overline{1,n}, \tag{2.4.20}$$

де $\underline{\Delta a_{ij}} = \max\{\underline{\gamma_{ij}}, -a_{ij}\}$, $\overline{\Delta a_{ij}} = \min\{\overline{\gamma_{ij}}, 1 - a_{ij}\}$, і

$$\underline{\Delta q_i} \le \Delta q_i \le \overline{\Delta q_i}, \quad i = \overline{1,n}, \tag{2.4.21}$$

де $\underline{\Delta q_i} = \max\{\underline{\gamma_i}, -q_i\}$ $\overline{q_i} = \min\{\overline{\gamma_i}, 1 - q_i\}$. Обмеження (2.4.20) випливають з об'єднання частини обмежень (2.4.12) і обмежень (2.4.14), а обмеження (2.4.21) – з об'єднання решти обмежень (2.4.12) і обмежень (2.4.15).

Задачі максимізації величини D відповідає оптимізаційна задача

$$f_1(\Delta A, \Delta q) = \frac{(q + \Delta q)^T (E - (A + \Delta A))^{-1} h}{1 - (q + \Delta q)^T (E - (A + \Delta A))^{-1} \alpha} \to \max, \tag{2.4.22}$$

де ΔA і Δq задовольняють обмеженням (2.4.17) – (2.4.21).

Задачі максимізації величини k (мультиплікатора «приріст доходів–приріст виробництва») відповідає оптимізаційна задача

$$f_2(\Delta A, \Delta q) = (q + \Delta q)^T (E - (A + \Delta A))^{-1} \alpha \to \max, \tag{2.4.23}$$

де ΔA і Δq задовольняють обмеженням (2.4.17) – (2.4.21).

Обидві оптимізаційні задачі є складними багатоекстремальними задачами математичного програмування. Багатоекстремальність

пов'язана тільки зі складними видами функцій $f_1(\Delta A, \Delta q)$ і $f_2(\Delta A, \Delta q)$ (сама ж область допустимих значень, задана нерівностями (2.4.17) – (2.4.21), є опуклою). Більш того, ці функції не при всіх ΔA і Δq є неперервними. Щоб виключити можливість існування точок розриву, потрібно виконання таких умов:

а) матриця $E-(A+\Delta A)$ є неособливою;

б) $(q+\Delta q)^T (E-(A+\Delta A))^{-1}\alpha \neq 1$.

Обидві умови повинні виконуватися для задачі з цільовою функцію $f_1(\Delta A, \Delta q)$, і тільки перша умова – для задачі з цільовою функцію $f_2(\Delta A, \Delta q)$.

Оскільки з нерівностей (2.4.17) випливає продуктивність матриці $A+\Delta A$, то умова а) виконується для ΔA, що належать як області допустимих рішень (2.4.17) – (2.4.21), так і її деякого, досить великого, околу. Збільшення обсягів виробництва буде вимагати додаткових матеріальних витрат навіть при використанні ефективних ресурсозберігаючих технологій, до того ж знову створена добавлена вартість ніколи не буде використана цілком тільки для споживання. Тому для реальних технологій завжди буде виконуватися $(q+\Delta q)^T (E-(A+\Delta A))^{-1}\alpha < 1$, що означає виконання умови б).

Звідси в області, де ΔA і Δq задовольняють обмеженню (2.4.17), функція $f_1(\Delta A, \Delta q)$ буде неперервно-диференційовною і її градієнт дорівнює

$$\frac{\partial f_1(\Delta A, \Delta q)}{\partial \Delta q_j} = \frac{1}{1-(q+\Delta q, \tilde{\alpha})}(e_j, \tilde{h}) + \frac{(q+\Delta q, \tilde{h})}{(1-(q+\Delta q, \tilde{\alpha}))^2}(e_j, \tilde{\alpha}), \quad j = \overline{1,n},$$

$$\frac{\partial f_1(\Delta A, \Delta q)}{\partial \Delta a_{ij}} = \frac{1}{1-(q+\Delta q, \tilde{\alpha})}(e_i, \tilde{q})(e_j, \tilde{h}) +$$

$$+\frac{(q+\Delta q, \tilde{h})}{(1-(q+\Delta q, \tilde{\alpha}))^2}(e_i, \tilde{q})(e_j, \tilde{\alpha}), i, j = \overline{1,n}.$$

Тут прийнято такі позначення: $\tilde{h} = (E-(A+\Delta A))^{-1}h$,

$\tilde{\alpha} = (E - (A + \Delta A))^{-1}\alpha$, $\tilde{q} = ((E - (A + \Delta A))^T)^{-1}(q + \Delta q)$, e_i – n-вимірний вектор, i-а компонента якого дорівнює одиниці, а всі інші дорівнюють нулю.

При обчисленні градієнтів для функції $f_1(\Delta A, \Delta q)$ по змінним Δa_{ij} використано правило диференціювання оберненої матриці X^{-1} ($X = \{x_{ij}\}_{ij}^{nn}$):

$$\frac{\partial X^{-1}}{\partial x_{ij}} = -X^{-1}\frac{\partial X}{\partial x_{ij}}X^{-1} = -X^{-1}e_i e_j^T X^{-1},$$

яке є наслідком диференціювання тотожності $XX^{-1} = E$.

Для функції $f_2(\Delta A, \Delta q)$ градієнти обчислюються трохи простіше, ніж для функції $f_1(\Delta A, \Delta q)$. Однак, проблеми в обох оптимізаційних задачах однакові. А саме, властивості цільових функцій створюють проблеми навіть при знаходженні локального екстремуму, що накладає певні вимоги на вибір алгоритмів для їхнього розв'язання. Із самою системою обмежень особливих проблем немає, хоча опуклі обмеження (2.4.19) і є негладкими.

Слід зазначити, що результати розрахунків, виконаних на основі наведених оптимізаційних моделей (максимізація цільових функцій (2.4.9) і (2.4.10) при обмеженнях у вигляді (2.4.11) – (2.4.16) або (2.4.17) – (2.4.21)) не можуть мати директивний характер. Вони повинні допомогти визначити бажану структуру виробничих технологій, що сприяє інтенсифікації соціально-економічного розвитку країни, виявити шляхи перетворення існуючої структури до бажаного, оцінити необхідні для цього ресурси й ін. Важливу роль відіграє аналіз зміни у часі технологічних коефіцієнтів a_{ij} за ряд останніх років, виявлення тенденцій наближення (або віддалення) реальних значень a_{ij} до бажаних значень, отриманих у результаті розрахунків у вищезгаданих моделях. З цього погляду одержання серії локальних екстремумів функцій (2.4.9) або (2.4.10) зі значеннями, достатньо близькими до їхніх глобальних максимумів, виглядає

кориснішим для наступного застосування за власне пошук глобального екстремуму.

2.4.5 Методи розв'язування задач максимізації D і k

Для знаходження локальних екстремумів у задачах (2.4.17) – (2.4.22) або (2.4.17) – (2.4.21), (2.4.23) була реалізована програма MULSTR на мові програмування RATFOR (препроцесор Фортрану). Ці задачі зводилися до задач безумовної максимізації, використовуючи метод негладких штрафних функцій. Для розв'язування останніх застосовувався $r(\alpha)$-алгоритм [5, 6], що зарекомендував себе практично ефективним методом при мінімізації опуклих недиференційовних функцій. По-перше, він дозволяє врахувати негладкість обмеження (2.4.19). По-друге, для урахування зазначених вище особливостей цільових функцій використовувалася модифікація [7] для того, щоб градієнти цільових функцій обчислювати тільки в області, де вона неперервно-диференційовна. Багатоекстремальність задач враховувалася шляхом проведення розрахунків з різними початковими точками (метод мультистарт). У програмі MULSTR передбачене керування такими параметрами: а) початковим наближенням; б) штрафним множником (використовується при зведенні задач до задачі безумовної оптимізації); в) параметрами $r(\alpha)$-алгоритму. До керуючих параметрів $r(\alpha)$-алгоритму віднесені такі: h_0 – початковий кроковий множник, $maxitn$ – максимальне число ітерацій; ε_x – точність для критерію зупинки по аргументу. Інші параметри $r(\alpha)$-алгоритму обрані у відповідності з рекомендаціями [8], тобто коефіцієнт розтягу простору – $\alpha = 2$; параметри адаптивного регулювання крокового множника – $q_1 = 0.95$, $q_2 = 1.1$, $n_h = 3$. При обчисленні градієнтів цільових функцій, де потрібно розв'язувати системи лінійних рівнянь, використані чисельно стійкі підпрограми DECOMP і SOLVE [9], які базуються на варіанті гаусового виключення з частковим вибором ведучого елемента.

З допомогою програми MULSTR були проведені тестові розрахунки на реальних даних для агрегованого 18-галузевого балансу. Метою експериментів було, по-перше, оцінити ефективність запропонованого алгоритму на реальних даних і при реальній розмірності задачі, по-друге, проаналізувати рекомендації для управлінських рішень, отриманих з її допомогою. У цих експериментах:

а) як коефіцієнти вектора α розглядалися показники питомої ваги галузей у компонентах кінцевого продукту, що залежать від доходів споживачів (індивідуального споживання та інвестицій).

б) в обмеженнях (2.4.17) було прийнято $\beta = 0.85$, що забезпечувало при наявній точності вхідних даних можливість збільшення частки доданої вартості в ціні галузей не менш, ніж 5 %.

в) допускалася зміна тільки тих коефіцієнтів прямих витрат (у межах 50 % від їхніх початкових значень), які більше 10^{-3}; інші вважалися незмінними. У цьому випадку число змінних задачі дорівнювало 200 (з них 182 – це зміни коефіцієнтів у матриці прямих витрат, а 18 – зміни компонентів вектора q).

г) з групи обмежень (2.4.19) розглядалося тільки одне обмеження на обсяг інвестицій, що залучаються для структурно-технологічних змін. Проводилися варіантні розрахунки при різних обсягах таких інвестицій.

Проведені розрахунки показали як можливість застосування агрегованих міжгалузевих моделей для планування структурно-технологічних змін, так і основні напрямки таких змін. Збільшення питомої ваги заробітної плати в ціні продукції більшості галузей повинно стати основою заходів щодо подолання економічної кризи, а скорочення матеріальних витрат створить передумови для такого збільшення без розвитку інфляційних процесів. Результати розрахунків, проведених на основі 18-галузевого балансу, показали як можливість і необхідність кардинальних структурно-технологічних змін в економіці України, так і напрямки таких змін – енергозбереження, зниження матеріалоємності виробництва, скорочення транспортних витрат і інше.

Надалі програма MULSTR була модифікована і застосовувалася для ряду тестових розрахунків для задач (2.4.17) – (2.4.22) і (2.4.17) – (2.4.21), (2.4.23), що включали 39 галузей. Варіант програми, який при цьому використовувався, включав знаходження 10 локальних екстремумів (старт із 10-ти початкових наближень). Як розв'язок задачі використовувався найкращий екстремум, знайдений з цих 10 точок. Керування початковими точками реалізовано за допомогою встановлення нижніх і верхніх границь на змінні ΔA і Δq. Ця програма використовувалася для аналізу задач (2.4.17) – (2.4.22) або (2.4.17) – (2.4.21), (2.4.23) при відсутності ресурсних обмежень у діалоговій системі.

Слід відзначити, що запропонований вище метод, реалізований програмою MULSTR, легко розпаралелити для багатопроцесорного комплексу. Розглянемо багатоекстремальну задачу

$$\min_{x \in M} f(x) \qquad (2.4.24)$$

де $f(x)$ – неперервна функція, що залежить від векторного аргументу, M – опукла обмежена множина з евклідового простору R^n. Нехай маємо k точок, які обираються у той чи інший спосіб з множини M. Найкращим мінімумом задачі (2.4.24) для заданого методу локального спуску будемо вважати найменший з локальних мінімумів задачі (2.4.24), що отримані за допомогою цього методу при послідовних стартах з кожної з вказаних точок.

Процес знаходження найкращого з розв’язків задачі (2.4.24) легко розпаралелити на p процесорах, якщо на кожному з них розв’язувати окремі локальні підзадачі (знаходження локального екстремуму з однієї початкової точки). Очевидно, що прискорення, яке при цьому можна одержати по відношенню до однопроцесорного комп'ютера, буде в кращому випадку лінійним, тобто $q =]k / p[+ 1$, де $]\cdot[$ – ціла частина. Якщо ж метод "локального" пошуку забезпечити інформацією про вже знайдене найкраще (рекордне) значення функції, то навіть в однопроцесорному випадку ми можемо отримати виграш. Його можна досягти за рахунок припинення роботи методу

локального пошуку, якщо стає зрозумілим, що рекордне значення функції не може бути досягнуто. У випадку p процесорів це може привести до істотного прискорення процесу знаходження найкращого з розв'язків задачі (2.4.24).

Відзначимо, що чисельні експерименти продемонстрували наступну особливість у тестових задачах. З різних початкових наближень була отримана серія розв'язків з майже однаковими значеннями цільової функції (відхилення порядку $0.3\,\% - 1\,\%$ від максимального значення в серії), але з істотними розходженнями кількох компонентів розв'язків. Наступне твердження до деякої міри пояснює цей ефект.

Теорема 2.4.1. Нехай $\left(\Delta A^{(1)}, \Delta q^{(1)}\right)$ і $\left(\Delta A^{(2)}, \Delta q^{(2)}\right)$ – довільні допустимі розв'язки задачі (2.4.17) – (2.4.22) і (2.4.17) – (2.4.21), (2.4.23), для яких покомпонентно виконується рівність:

$$\left(E - \left(A + \Delta A^{(1)}\right)^T\right)^{-1} \left(q + \Delta q^{(1)}\right) =$$

$$= \left(E - \left(A + \Delta A^{(2)}\right)^T\right)^{-1} \left(q + \Delta q^{(2)}\right). \qquad (2.4.25)$$

Нехай також

$$\left(\Delta A_\lambda, \Delta q_\lambda\right) = \lambda\left(\Delta A^{(1)}, \Delta q^{(1)}\right) + (1-\lambda)\left(\Delta A^{(2)}, \Delta q^{(2)}\right),$$

де $0 < \lambda < 1$ – довільне число. Тоді

$$f_i\left(\Delta A^{(1)}, \Delta q^{(1)}\right) = f_i\left(\Delta A_\lambda, \Delta q_\lambda\right) = f_i\left(\Delta A^{(2)}, \Delta q^{(2)}\right),\ i = \overline{1,2}$$

і $\left(\Delta A_\lambda, \Delta q_\lambda\right)$ буде допустимим розв'язком вищезгаданих задач.

Доведення теореми 2.4.1 базується на твердженні, що розглядається в наступному підрозділі і буде приведено після нього. Поки ж зупинимося на деяких висновках з цієї теореми .

Допустимі точки з однаковими значеннями цільової функції, що належать множині, яка задовольняє (2.4.25), утворюють структури, подібні сукупності відрізків прямих. Якщо крайні точки одного з таких відрізків відрізняються лише кількома компонентами, то усі

внутрішні точки відрізка також будуть розрізнятися між собою тільки значеннями цих компонентів. Якщо мова йде про значення, близькі до глобального максимуму, результатом роботи *r*-алгоритму для сукупності початкових точок буде одна з таких структур, що і відповідає результатам чисельних експериментів.

2.4.6 Розширені оптимізаційні задачі

Введемо нові змінні

$$z = \left(E - \left(A + \Delta A\right)^T\right)^{-1} \left(q + \Delta q\right) \tag{2.4.26}$$

і з їхньою допомогою побудуємо розширені оптимізаційні задачі для задач (2.4.17) – (2.4.22) і (2.4.17) – (2.4.21), (2.4.23). Враховуючи, що з (2.4.26) випливає $z^T = \left(q + \Delta q\right)^T \left(E - \left(A + \Delta A\right)\right)^{-1}$, цільові функції $f_1\left(\Delta A, \Delta q\right)$ і $f_2\left(\Delta A, \Delta q\right)$ приймуть досить простий вигляд. Так цільова функція для величини D буде мати такий дробово-лінійний вигляд:

$$F_1(z) = \frac{z^T h}{1 - z^T \alpha}, \tag{2.4.27}$$

а цільова функція для мультиплікатора «приріст доходів–приріст виробництва» k стане лінійної, тобто

$$F_2(z) = z^T \alpha. \tag{2.4.28}$$

В результаті введення змінних z до обмежень задач додадуться обмеження, які у матричній формі можна записати

$$z - \left(A + \Delta A\right)^T z = q + \Delta q. \tag{2.4.29}$$

Обмеження (2.4.29) є результатом множення обох частин у співвідношенні (2.4.26) на матрицю $\left(E - \left(A + \Delta A\right)^T\right)$. Тоді задачі (2.4.17) – (2.4.22) буде відповідати розширена задача оптимізації в такій формі:

$$F_1(z) = \frac{z^T h}{1 - z^T \alpha} \to \max \tag{2.4.30}$$

за обмежень

$$z_j - \sum_{i=1}^{n}\left(a_{ji} + \Delta a_{ji}\right) z_i = q_j + \Delta q_j, \quad j = \overline{1,n}, \tag{2.4.31}$$

$$\beta\left(a_{jj} + \Delta a_{jj}\right) + \beta\left(l_j\left(q_j + \Delta q_j\right) + d_j\right) +$$

$$+ \sum_{i=1, i\neq j}^{n}\left(a_{ij} + \Delta a_{ij}\right) \le \beta, \quad j = \overline{1,n}, \tag{2.4.32}$$

$$a_{jj} + \Delta a_{jj} + l_j\left(q_j + \Delta q_j\right) + d_j \le 1, \quad j = \overline{1,n}, \tag{2.4.33}$$

$$\sum_{j=1}^{n}\sum_{i=1}^{n}\left(b_{kij}^{-} \max\left(0, -\Delta a_{ij}\right) + b_{kij}^{+} \max\left(0, \Delta a_{ij}\right)\right) \le B_k, \quad k = \overline{1,K}, \tag{2.4.34}$$

$$\underline{\Delta q_i} \le \Delta q_i \le \overline{\Delta q_i}, \quad \underline{\Delta a_{ij}} \le \Delta a_{ij} \le \overline{\Delta a_{ij}}, \quad i, j = \overline{1,n}. \tag{2.4.35}$$

Тут обмеження (2.4.31) є записом матричних обмежень (2.4.29) для компонентів матриці ΔA, вектора Δq і вектора z. Обмеження (2.4.35) є об'єднанням обмежень, пов'язаних з нижніми і верхніми границями на компоненти вектора Δq і матриці ΔA. Інші обмеження такі ж, як і в задачах (2.4.17) – (2.4.21), (2.4.23). Аналогічний вигляд прийме і задача (2.4.17) – (2.4.21), (2.4.23) з тією лише відмінністю, що їй буде відповідати максимізація цільової функції у формі (2.4.28), тобто

$$F_2(z) = z^T \alpha \to \max. \tag{2.4.36}$$

Для обох задач, що розглядаються, справедливе таке твердження.

Теорема 2.4.2. Нехай $x^{(1)} = (\Delta A^{(1)}, \Delta q^{(1)}, z^*)$ та $x^{(2)} = (\Delta A^{(2)}, \Delta q^{(2)}, z^*)$ – довільні допустимі розв'язки задач (2.4.31) – (2.4.36) і (2.4.30) – (2.4.35) з однаковими значеннями змінних $z = z^*$. Нехай також $x(\lambda) = \lambda x^{(1)} + (1-\lambda)x^{(2)}$, де $0 < \lambda < 1$ – довільне число. Тоді значення функцій $F_i(z)$, $i = \overline{1,2}$, будуть співпадати для розв'язків $x^{(1)}$, $x^{(2)}$ і $x(\lambda)$, а $x(\lambda)$ буде допустимим розв'язком для довільного $0 < \lambda < 1$.

Доведення. Покажемо, що для довільного $0 < \lambda < 1$ точка $x(\lambda)$ буде задовольняти обмеженням (2.4.31) – (2.4.35). Дійсно, обмеження (2.4.32), (2.4.33), (2.4.35) є лінійними нерівностями, а ліва частина обмежень (2.4.34) є опуклою функцією. Тому ці обмеження задають опуклу множину. Якщо точки $x^{(1)}$ і $x^{(2)}$ належать цій множині, то і їхня опукла комбінація $x(\lambda)$ також буде належати йому, тобто вона задовольняє співвідношенням (2.4.32) – (2.4.35). Далі, у силу допустимості $x^{(1)}$ і $x^{(2)}$, справедливі рівності

$$(E - (A + \Delta A^{(1)})^T)z^* = q + \Delta q^{(1)}, \ (E - (A + \Delta A^{(2)})^T)z^* = q + \Delta q^{(2)}.$$

Помноживши перше з цих рівностей на λ, а друге на $(1-\lambda)$ і склавши їх, отримаємо

$$(E - (A + \lambda\Delta A^{(1)} + (1-\lambda)\Delta A^{(2)})^T)z^* = q + \lambda\Delta q^{(1)} + (1-\lambda)\Delta q^{(2)}. \quad (2.4.37)$$

Звернувши увагу на те, що $z_\lambda = \lambda z^* + (1-\lambda)z^* = z^*$, рівність (2.4.37) можна переписати у вигляді

$$(E - (A + \Delta A_\lambda)^T)z^* = q + \Delta q_\lambda, \quad (2.4.38)$$

де $(\Delta A_\lambda, \Delta q_\lambda, z_\lambda)$ – компоненти розв'язку $x(\lambda)$. Згідно (2.4.38) точка $x(\lambda)$ задовольняє рівності (2.4.31) і є допустимим розв'язком задач (2.4.36), (2.4.31) – (2.4.35) і (2.4.30) – (2.4.35).

Відзначимо, що значення змінних z для розв'язку $x(\lambda)$ співпадають зі значеннями цих змінних для розв'язків $x^{(1)}$ і $x^{(2)}$. Оскільки цільові функції $\overline{f}_1(z)$ і $\overline{f}_2(z)$ цих задач залежать тільки від z, їхні значення у точках $x^{(1)}$, $x^{(2)}$ і $x(\lambda)$ також співпадуть, звідки випливає справедливість теореми.

Довівши теорему 2.4.2, повернемося до доведення теореми 2.4.1 з попереднього підрозділу.

Відзначимо, що з допустимості розв'язків $\left(\Delta A^{(1)}, \Delta q^{(1)}\right)$ і $(\Delta A^{(2)}, \Delta q^{(2)})$ випливає існування матриць $(E-(A+\Delta A^{(i)})^T)^{-1}$, $i=\overline{1,2}$. Кожному допустимому розв'язку $(\Delta A^{(i)}, \Delta q^{(i)})$, $i=\overline{1,2}$, задач (2.4.17) – (2.4.22) або (2.4.17) – (2.4.21), (2.4.23) буде відповідати єдиний допустимий розв'язок $(\Delta A^{(i)}, \Delta q^{(i)}, z^{(i)})$ задач (2.4.36), (2.4.31) – (2.4.35) і (2.4.30) – (2.4.35), при цьому в силу (2.4.25) і (2.4.31) буде виконуватися $z^{(1)} = z^{(2)} = z^*$.

Матриця $E-(A+\Delta A_\lambda)$ також буде неособливою і допустимому розв'язку $(\Delta A_\lambda, \Delta q_\lambda)$ задач (2.4.17) – (2.4.22) або (2.4.17) – (2.4.21), (2.4.23) також відповідає єдиний допустимий розв'язок $(\Delta A_\lambda, \Delta q_\lambda, z_\lambda)$ перетворених задач. Тому можна застосувати теорему 2.4.2 для $x^{(1)} = (\Delta A^{(1)}, \Delta q^{(1)}, z^{*)})$, $x^{(2)} = (\Delta A^{(2)}, \Delta q^{(2)}, z^*)$ і $x(\lambda) = (\Delta A_\lambda, \Delta q_\lambda, z_\lambda)$. Відповідно до цієї теореми, $F_i(z^*) = F_i(z_\lambda)$, $i=\overline{1,2}$. У той же час, в силу еквівалентності відповідних задач у їх початковому і перетвореному вигляді, повинно виконуватися $f_i(\Delta A^{(i)}, \Delta q^{(i)}) = F_i(z^*)$, $i=\overline{1,2}$. Звідси $f_i(\Delta A^{(i)}, \Delta q^{(i)}) = f_i(\Delta A_\lambda, \Delta q_\lambda)$, $i=\overline{1,2}$, що і доводить справедливість теореми.

Відповідно до теореми 2.4.2 множина X_1 значень змінних ΔA і Δq, що будуть допустимими при заданих значеннях z, буде опуклою. Це спрощує побудову даної множини.

Реалізуючи з декількох початкових точок процедуру субградієнтного методу з перетворенням простору, який припускає

відомим значення цільової функції [10] (а $\overline{f}_i(z)$, $i=\overline{1,2}$, відомі, якщо z задано), одержимо кілька точок вищезгаданої множини. Їх опукла лінійна комбінація також буде, за теоремою 2.4.2, належати даній множині. Конструктивні алгоритми перебору різних точок з X_1 можуть бути також побудовані на основі субградієнтних методів, що використовують зовнішню апроксимацію множини екстремумів еліпсоїдами, наприклад, методу знаходження допустимої точки системи опуклих нерівностей [11], який забезпечує прискорену збіжність до граничних точок множини X_1. При цьому апроксимацію X_1 еліпсоїдом, що побудована на попередньому кроці зазначеного методу, можна використовувати для знаходження чергової граничної точки X_1.

Зазначимо, що компоненти вектора z відображають структуру кінцевих доходів, отриманих від різних видів економічної діяльності. Дана структура визначає паритет інтересів (існуючий або бажаний) між різними суб'єктами господарювання. Проведення варіантних розрахунків (при різних z) і порівняння отриманих значень критеріїв з розв’язками задач виду (2.4.17) – (2.4.22) і (2.4.17) – (2.4.21), (2.4.23) дозволить оцінити ступінь впливу приватних інтересів на ефективність розвитку економіки і спрогнозувати, чи буде позитивно сприйнято суб'єктами господарювання запропоноване (глобально оптимальне або близьке до нього) рішення або ж воно зустріне опір. Моделі (2.4.31) – (2.4.36) і (2.4.30) – (2.4.35), розглянуті при заданих z, є моделями з фіксованими цілями, в яких результати визначені, але потрібно побудувати множину інструментів, тобто дій, що дозволяють

досягти ці результати. Застосування таких моделей відіграє важливу роль як для попереднього аналізу ситуації, так і для остаточного відбору раніше отриманих рішень. З цього погляду генерація множини допустимих розв'язків задач (2.4.31) – (2.4.36) і (2.4.30) – (2.4.35) зі значеннями цільових функцій, досить близькими до глобального максимуму, і проведення розрахунків у діалоговому режимі відіграє важливу роль [12].

Порівнюючи між собою підходи до проведення модельних розрахунків, викладені в даному і попередньому розділах, слід зазначити такі їхні переваги і недоліки.

Підхід, що базується на розв'язанні задач (2.4.17) – (2.4.22) і (2.4.17) – (2.4.21), (2.4.23) без їхнього перетворення, дозволяє істотно зменшити розмірність задачі. Оскільки всі обмеження мають форму нерівностей, точний розв'язок задачі може бути отриманий при обмежених значеннях штрафних коефіцієнтів, що обмежує яружність функції, яка максимізується. Множина допустимих розв'язків задачі буде опуклою, що в ряді випадків полегшує знаходження першого допустимого розв'язку. У той же час можливості аналізу отриманих розв'язків при даному підході обмежені.

Підхід, викладений у даному підрозділі, призводить до деякого збільшення розмірності розв'язуваної задачі, і можливості скорочення розмірності будуть обмежені. Множина допустимих розв'язків задачі стає неопуклою, а наявність нелінійних (квадратичних) обмежень-рівностей вимагає великих (теоретично – необмежених) значень відповідних їм штрафних коефіцієнтів, що збільшує яружність штрафної функції. У той же час у розширеній задачі не

використовуються не усюди визначені функції і виключені проблеми, пов'язані з виходом за область їхнього визначення при пошуку розв’язку задачі. Запропонований підхід дає широкі можливості для аналізу отриманих розв’язків, розгляду моделей як задач з фіксованими цілями і дозволяє оцінити значення цільової функції в точці її глобального максимуму.

Таким чином, обидва підходи мають переваги і недоліки, вони доповнюють один одного. Тому доцільне їхнє одночасне застосування при проведенні модельних розрахунків.

Для знаходження локальних екстремумів в розширених задачах (2.4.31) – (2.4.36) і (2.4.30) – (2.4.35) реалізована програма MULSTR1 на мові програмування RATFOR (препроцесор Фортрану); її код наведено у Додатку А. Вона побудована подібно програмі MULSTR, тобто задачі (2.4.31) – (2.4.36) і (2.4.30) – (2.4.35) зводяться до задач безумовної максимізації, використовуючи метод негладких штрафних функцій, а вже до останніх застосовується $r(\alpha)$-алгоритм. Програма MULSTR1 є значно простіша за програму MULSTR з огляду на те, що при обчисленні градієнтів цільових функцій тут не потрібно розв’язувати системи лінійних рівнянь.

У програмі MULSTR1 передбачене керування трьома штрафними множниками (використовуються при зведенні задач (2.4.31) – (2.4.36) і (2.4.30) – (2.4.35) до задачі безумовної оптимізації). Кожний зі штрафних множників пов'язаний відповідно зі своєю групою обмежень. Так, перший штрафний множник відповідає групі обмежень (2.4.31), другий – групам обмежень (2.4.32) і (2.4.33), пов'язаних з коефіцієнтом β, і третій – для групи ресурсних обмежень (2.4.34).

Обмеження (2.4.35) у вигляді двосторонніх обмежень на ΔA і Δq не включаються в негладку штрафну функцію, а враховуються за допомогою заміни змінних ΔA і Δq на нові змінні, які визначені на всьому евклідовому просторі. Цей прийом часто застосовується при використанні r-алгоритмів і інтерпретується як "парне" періодичне продовження функції, заданої на відрізку [5]. До керуючих параметрів $r(\alpha)$-алгоритму віднесені: h_0 – початковий кроковий множник, $maxitn$ – максимальне число ітерацій; ε_x – точність для критерію зупинки по аргументі, ε_g – точність для критерію зупинки по нормі субградієнта, α – коефіцієнт розтягу простору; q_1, q_2, n_h – параметри адаптивного регулювання крокового множника. Багатоекстремальність розширених задач (2.4.31) – (2.4.36) і (2.4.30) – (2.4.35) враховувалася шляхом проведення розрахунків з 10 різних початкових точок.

2.4.7 Тестові експерименти (Mulstr1)

Приведемо результати тестових експериментів, отримані за допомогою програми Mulstr1 для розширених оптимізаційних задач (2.4.31) – (2.4.36) і (2.4.30) – (2.4.35). Розглядалося 7 галузей, для яких матриця коефіцієнтів прямих витрат (матриця A) і частка оплати праці в ціні продукції (вектор q) такі:

$$A = \begin{pmatrix} 0.337 & 0.139 & 0.215 & 0.127 & 0.146 & 0.112 & 0.1960 \\ 0.023 & 0.251 & 0.179 & 0.089 & 0.019 & 0.131 & 0.0050 \\ 0.163 & 0.176 & 0.191 & 0.097 & 0.103 & 0.095 & 0.0870 \\ 0.012 & 0.009 & 0.157 & 0.031 & 0.029 & 0.026 & 0.0940 \\ 0.009 & 0.010 & 0.008 & 0.226 & 0.107 & 0.006 & 0.0071 \\ 0.153 & 0.121 & 0.099 & 0.031 & 0.025 & 0.019 & 0.0330 \\ 0.161 & 0.193 & 0.103 & 0.101 & 0.095 & 0.087 & 0.0910 \end{pmatrix}, \quad q = \begin{pmatrix} 0.05 \\ 0.02 \\ 0.01 \\ 0.08 \\ 0.09 \\ 0.12 \\ 0.14 \end{pmatrix}.$$

Мультиплікатор витрат на оплату праці дорівнює $l = (1.320, 1.000, 1.375, 1.375, 1.375, 1.375, 1.375)$, а частка інших складової доданої вартості – $d = (0.01, 0.05, 0.01, 0.05, 0.10, 0.10, 0.15)$. Структура індивідуального споживання є $\alpha = (0.15, 0.05, 0.05, 0.15, 0.20, 0.20, 0.20)$, а структура колективного споживання – $h = (0.1, 0.2, 0.2, 0.1, 0.05, 0.05, 0.3)$. Кількість ресурсів, що використовуються для структурно-технологічних змін дорівнює 1, і загальна кількість цього ресурсу $B_1 = 3.5$. Коефіцієнти питомих витрат ресурсів $\{b_{ij1}\}_{ij}^{nn}$ задані матрицею

$$\{b_{ij1}\}_{ij}^{nn} = \begin{pmatrix} 3.00 & 3.00 & 3.00 & 1.00 & 1.00 & 2.00 & 0.50 \\ 3.00 & 3.00 & 3.00 & 1.00 & 1.00 & 2.00 & 0.50 \\ 3.00 & 3.00 & 3.00 & 1.00 & 1.00 & 2.00 & 0.50 \\ 3.00 & 3.00 & 3.00 & 1.00 & 1.00 & 2.00 & 0.50 \\ 3.00 & 3.00 & 3.00 & 1.00 & 1.00 & 2.00 & 0.50 \\ 3.00 & 3.00 & 3.00 & 1.00 & 1.00 & 2.00 & 0.50 \\ 3.00 & 3.00 & 3.00 & 1.00 & 1.00 & 2.00 & 0.50 \end{pmatrix}$$

Для цього прикладу досліджувалися найкращі значення цільових функцій у розширених оптимізаційних задачах (2.4.31) – (2.4.36) і (2.4.30) – (2.4.35) в залежності від трьох параметрів, що головним чином звужують множину допустимих розв'язків для цих задач. Перший з них коефіцієнт β, що задає деякий запас для запобігання інфляції витрат під впливом внутрішніх факторів. Другий –

двосторонні границі на змінні ΔA і Δq, а третій – значення правої частини ресурсного обмеження B_1. У таблицях 2.4.1 і 2.4.2 відображена динаміка зміни значень цільових функцій: F_1^* – для розширеної оптимізаційної задачі (2.4.31) – (2.4.36) і F_2^* – для розширеної оптимізаційної задачі (2.4.30) – (2.4.35). Розглядалися наступні значення коефіцієнта β: $\beta = 1$ (відсутність інфляції витрат під впливом внутрішніх факторів), $\beta = 0,95$ і $\beta = 0,9$. У таблиці 2.4.1 відображено динаміку зміни оптимальних доходів ($D^* = F_1^*$) і мультиплікатора «приріст доходів – приріст виробництва» ($k^* = F_2^*$) при різних можливих діапазонах зміни коефіцієнтів матриці A. Зміни розглядалися у відсотках від значень коефіцієнтів матриці від 50 % до 10 %. У таблиці 2.4.2 відображена динаміка оптимального доходу D^* і оптимального мультиплікатора «приріст доходів – приріст виробництва» k^* при різних значеннях правої частини ресурсного обмеження. Можливий діапазон зміни коефіцієнтів матриці А тут склав 50 % від їхніх початкових значень. З наведених нижче таблиць легко бачити, що звужуючи множину допустимих розв'язків задач (2.4.31) – (2.4.36) і (2.4.30) – (2.4.35) ми завжди погіршуємо (а іноді, і значно) основні макроекономічні показники для моделі міжгалузевого балансу – сукупний доход і мультиплікатор Кейнса.

Таблиця 2.4.1. Залежність D^* і k^* від діапазонів зміни коефіцієнтів матриці A

	$\beta = 1$		$\beta = 0,95$		$\beta = 0,9$	
%	D^*	k^*	D^*	k^*	D^*	k^*
50	2,16188	0,675099	1,88599	0,646365	1,67095	0,620267
40	2,03471	0,663156	1,78670	0,635206	1,57472	0,607878
30	1,90212	0,649541	1,65749	0,619389	1,43671	0,587395
20	1,75540	0,633151	1,47255	0,594165	1,21586	0,550846
10	1,57512	0,610443	1,23925	0,556535	0,95043	0,496638

Таблиця 2.4.2. Залежність D^* і k^* від значення правої частини ресурсного обмеження

	$\beta = 1$		$\beta = 0{,}95$		$\beta = 0{,}9$	
B_1	D^*	k^*	D^*	k^*	D^*	k^*
3,5	2,16188	0,675099	1,88599	0,646365	1,67095	0,620267
3,0	2,13116	0,672061	1,83608	0,639725	1,60787	0,613146
2,5	2,09221	0,666546	1,75850	0,6319	1,52247	0,599653
2,0	2,00178	0,657497	1,66304	0,618496	1,39468	0,584259
1,5	1,88854	0,645884	1,53982	0,601932	1,25711	0,562413
1,0	1,75009	0,627479	1,3929	0,582262	1,09802	0,534037

2.4.8 Міжгалузева модель структурно-технологічних змін з індексами цін

Незважаючи на те, що можливості держави керувати цінами обмежені в умовах ринкової економіки, цінове регулювання залишається діючим інструментом економічної політики. Зокрема, воно може бути використане для забезпечення структурно-технологічних перетворень, стимулювання енерго- і ресурсозбережень. У зв'язку з цим існує необхідність у розробці моделей, подібних (2.4.9), (2.4.11) – (2.4.16) і (2.4.10) – (2.4.16), в яких керованими параметрами, що впливають на коефіцієнти прямих витрат, будуть ціни на продукцію галузей. Розглянемо одну з таких моделей [2].

Подібно до раніше розглянутих задач припустимо, що економіку країни утворюють n галузей. Позначимо a_{ij}^0, $i, j = \overline{1,n}$, коефіцієнти прямих витрат у вартісному вимірі, які розраховані для цін, що діяли на момент складання міжгалузевого балансу. Необхідно визначити доцільні зміни цих цін, максимізуючи величину кінцевих доходів споживачів або мультиплікатор.

З огляду на постановку задачі, позначимо p_i індекс зміненої ціни на продукцію галузі i ($i=\overline{1,n}$) по відношенню до ціни, що діяла на момент складання балансу; q_i позначимо частину доходів споживачів у ціні продукції галузі i. Остання величина виражається в одиницях початкової ціни даної продукції. Подібно до попередніх моделей припустимо, що величина $\tilde{q}_i$ доданої вартості у вартості одиниці продукції галузі i лінійно залежить від q_i:

$$\tilde{q}_i = l_i q_i + d_i,\ i=\overline{1,n},$$

де $l_i > 0$ і $q_i > 0$ – деякі задані коефіцієнти, зміст яких той же, що і для попередніх моделей.

З рівнянь міжгалузевого балансу для цін [13]:

$$p_j = \sum_{i=1}^{n} a_{ij}^0 p_i + \tilde{q}_j,\ j=\overline{1,n},$$

випливають співвідношення

$$p_j = \sum_{i=1}^{n} a_{ij}^0 p_i + l_j q_j + d_j,\ j=\overline{1,n}, \qquad (2.4.39)$$

утворюючу першу групу обмежень розглянутої моделі.

Коефіцієнти прямих витрат $a_{ij}(p)$ для діючих цін, що залежать від цінових індексів $p=(p_1,\ldots,p_n)$, визначаються з a_{ij}^0 у такий спосіб:

$$a_{ij}(p) = \frac{a_{ij}^0 p_i}{p_j},\ i,j=\overline{1,n}. \qquad (2.4.40)$$

Сума цінових індексів після зміни цін повинна, відповідно до закону вартості, залишитися незмінною. Оскільки для цін балансу виконувалося $p_i = 1$, $i=\overline{1,n}$, поточні значення p_i повинні задовольняти обмеженню

$$\sum_{i=1}^{n} p_i = n. \qquad (2.4.41)$$

Значення змінних моделі – p_i і q_i – повинні бути невід'ємними:

$$p_i \geq 0,\ i=\overline{1,n},\ q_i \geq 0,\ i=\overline{1,n}. \qquad (2.4.42)$$

Серед значень $p=(p_1,\ldots,p_n)$ і $q=(q_1,\ldots,q_n)$, що задовольняють умовам (2.4.39) – (2.4.42), потрібно знайти такі, які максимізували б величину кінцевих доходів споживачів

$$\tilde{f}_1(p,q)=\frac{q^T\left(E-A(p)\right)^{-1}h}{1-q^T\left(E-A(p)\right)^{-1}\alpha}\to\max \qquad (2.4.43)$$

або величину мультиплікатора «приріст доходів–приріст виробництва»

$$\tilde{f}_2(p,q)=q^T\left(E-A(p)\right)^{-1}\alpha^1\to\max. \qquad (2.4.44)$$

У (2.4.43) і (2.4.44) використовуються ті ж позначення, що й у (2.4.9), (2.4.10), $A(p)=\left\{a_{ij}(p)\right\}$.

Для того, щоб задачі (2.4.39) – (2.4.43) і (2.4.39) – (2.4.42), (2.4.44) були коректно сформульовані, необхідно, щоб $\left(E-A(p)\right)^{-1}$ існувала в будь-якій точці множини допустимих розв'язків. Для цього можна скористатися необхідною і достатньою умовою існування такої оберненої матриці [13]:

$$\lambda\left(A(p)\right)<1, \qquad (2.4.45)$$

де $\lambda\left(A(p)\right)$ – число Фробеніуса матриці $A(p)$ (дійсне додатне власне число цієї матриці, модуль якого буде максимальним серед усіх власних чисел).

З огляду на неточність вихідних даних і необхідність створення резервів при прийнятті управлінських рішень, доцільно замість (2.4.48) як обмеження моделі розглядати умову:

$$\lambda\left(A(p)\right)\leq\beta_1, \qquad (2.4.46)$$

де $0<\beta_1<1$ – деяке, заздалегідь задане число, достатньо близьке до 1. Крім умови (2.4.46) можна використовувати достатні умови існування $\left(E-A(p)\right)^{-1}$, наприклад:

$$\sum_{i=1}^{n} a_{ij}(p) < 1,\ j = \overline{1,n} \quad \text{або} \quad \sum_{i=1}^{n} a_{ij}(p) \le \beta_1 < 1,\ j = \overline{1,n}. \quad (2.4.47)$$

Умови (2.4.47) задають більш вузьку область розв'язків у порівнянні з (2.4.46), однак, будучи лінійними нерівностями, вони спрощують задачу.

До складу моделі також можуть бути включені двосторонні обмеження на можливі діапазони змін p_i і q_i:

$$\underline{p_i} \le p_i \le \overline{p_i},\ \underline{q_i} \le q_i \le \overline{q_i},\ i = \overline{1,n}. \quad (2.4.48)$$

Отримані при цьому оптимізаційні задачі мають неопуклі цільові функції, а якщо враховувати умову (2.4.46) – також і нелінійне обмеження. Для їхнього розв'язання можуть застосовуватися підходи, розглянуті в попередніх підрозділах. У зв'язку з наявністю обмежень-рівностей виникає питання сумісності обмежень (2.4.39) і (2.4.41). В принципі, ці обмеження повинні виконуватися для $p_i = 1$, $i = \overline{1,n}$, і для значень q_i, що відповідають частині доходів споживачів у ціні продукції галузей на момент складання балансу. Однак, у зв'язку зі зміною параметрів моделі, наприклад, l_i і d_i, $i = \overline{1,n}$, що залежать від фіскальної політики, це виконання може бути порушено і виникне необхідність швидкої перевірки сумісності зазначених обмежень. Це може бути виконано у такий спосіб. З (2.4.39) випливає, що

$$p = \left(E - A^T\right)^{-1}\left(\langle l, q\rangle + d\right), \quad (2.4.49)$$

де $A = \left\{a_{ij}^0\right\}$, $\langle l, q\rangle$ – покомпонентний добуток векторів $l = (l_1, \ldots, l_n)$ і $q = (q_1, \ldots, q_n)$, $d = (d_1, \ldots, d_n)$. Згідно (2.4.49) залежність p від q буде неперервною. Компоненти матриці $\left(E - A^T\right)^{-1}$ (коефіцієнти повних витрат), всі l_i і d_i будуть невід'ємними. Тому для існування такого q, при якому $p = (p_1, \ldots, p_n)$, визначені згідно (2.4.49), будуть задовольняти (2.4.41), необхідно і достатньо, щоб виконувалося

$$\tilde{e}^T \left(E - A^T \right)^{-1} d \le n$$

і існували б такі допустимі значення $q = q^1$, для яких би виконувалося

$$\tilde{e}^T \left(E - A^T \right)^{-1} \left(\left\langle l, q^1 \right\rangle + d \right) \ge n .$$

Тут $\tilde{e}$ – n-вимірний вектор, що складається з одиниць.

Алгоритм для розв'язування задачі (2.4.39) – (2.4.44) аналогічний алгоритму для розширених задач (2.4.31) – (2.4.36) і (2.4.30) – (2.4.35), і код його програмної реалізації MULSTR2 на мові RATFOR (препроцесор Фортрану), наведено у Додатку Г.

2.4.9 Модифікована міжгалузева модель структурно-технологічних змін

У даному підрозділі розглянуто модель, яка є подальшим розвитком моделей (2.4.9), (2.4.11) – (2.4.16) і (2.4.10) – (2.4.16). Вона відрізняється від раніше розглянутих моделей такими аспектами:

1) у моделі передбачені обмеження на наявні паливно-енергетичні ресурси і враховується можливість зміни питомих витрат цих ресурсів на виробництво продукції галузей;
2) передбачається, що структурно-технологічні зміни в деяких галузях (насамперед, в електроенергетиці) можуть здійснюватися двома шляхами:
    а) шляхом зміни інтенсивності використання наявних технологій виготовлення кінцевої продукції галузі (наприклад, збільшення питомої ваги теплоенергетики за рахунок зменшення питомої ваги ядерної енергетики або навпаки);
    б) шляхом зміни структури витрат для наявних технологій внаслідок упровадження технічних і технологічних інновацій. Впровадження нових технологій можна розглядати як окремий випадок першого з цих шляхів, зважаючи на те, що інтенсивність використання відповідної технології до

початку її впровадження дорівнювала нулеві, а після впровадження стала позитивним числом;

3) в моделі враховані екологічні обмеження, у тому числі й обмеження на викид парникових газів.

Модель є статичною.

Розглянемо основні припущення і введемо позначення моделі.

Нехай економіка складається з n чистих галузей, в яких можливі структурно-технологічні зміни; для галузі j_0 (якщо це електроенергетика, то по існуючій номенклатурі міжгалузевого балансу $j_0 = 17$) ці зміни деталізуються як зміни інтенсивності використання технологій і як зміни структури витрат для технологій. Позначимо K_{j_0} кількість технологій, що використовуються в галузі j_0, номери цих технологій позначимо k, $k = \overline{1, K_{j_0}}$. Нехай $W_k^{(j_0)}$ – інтенсивність використання технології k у галузі j_0; причому

$$\sum_{k=1}^{K_{j_0}} W_k^{(j_0)} = 1.$$

Позначимо a_{ij} коефіцієнти прямих витрат продукції галузі і на виготовлення одиниці продукції галузі j (коефіцієнти матриці Леонтьєва); при цьому $i = \overline{1,n}$, $j = \overline{1,n}$, $j \neq j_0$. Позначимо $\bar{a}_{ij_0}^k$ – питомі витрати продукції галузі i на виготовлення одиниці продукції галузі j_0 за технологією k. Відповідні елементи стовпця j_0 матриці Леонтьєва визначаються в такий спосіб:

$$a_{ij_0} = \sum_{k=1}^{K_{j_0}} \bar{a}_{ij_0}^k W_k^{(j_0)}. \qquad (2.4.50)$$

Позначимо q_j – частку кінцевих доходів у ціні продукції галузі j, $j = \overline{1,n}$. Припустимо, що частка доданої вартості $\bar{q}_j$ в ціні продукції галузі j лінійно залежить від q_j:

$$\overline{q}_j = l_j q_j + d_j, \quad j = \overline{1,n}, \tag{2.4.51}$$

де відомі коефіцієнти l_j і q_j відображають величину фіскального мультиплікатора кінцевих доходів i частку інших складової доданої вартості в ціні продукції галузі j. Припустимо, що обсяги кінцевого споживання y_i кожної з галузей лінійно залежать від доходу кінцевих споживачів у системи, яка моделюється:

$$y_i = \alpha_i D + h_i, \quad i = \overline{1,n}, \tag{2.4.52}$$

де α_i та h_i – також деякі відомі коефіцієнти.

Метою моделювання, подібно до задачі (2.4.9), (2.4.11) – (2.4.16), є визначення шляхів структурно-технологічних змін, що підвищують ефективність економіки. З урахуванням цього модель має кілька функцій мети: при проведенні розрахунків мова може йти про однокритеріальну оптимізацію, коли визначається екстремум однієї функції мети, відібраної з усіх інших, або ж про багатокритеріальну задачу, коли враховуються відразу всі функції мети. Для розв'язування такої задачі може використовуватися комбінація методів згорток і обмежень, розглянута в [3].

Позначимо g_j – вектор прямих питомих витрат обмежених ресурсів у галузі j, $j = \overline{1,n}$. При цьому величина g_{j_0} визначається з питомих витрат ресурсів для окремих технологій $g_{ij_0}^k$ та інтенсивності використання технологій $W_k^{(j_0)}$ подібно (2.4.50):

$$g_{j_0} = \sum_{k=1}^{K_{j0}} g_{j_0}^k W_k^{(j_0)}. \tag{2.4.53}$$

Нехай G – вектор наявних у наявності ресурсів. Матрицю, що складається зі стовпців g_j, $j = \overline{1,n}$, надалі будемо позначати g.

Позначимо $\tilde{g}_j$ – вектор питомих викидів забруднюючих речовин для галузі j. Вектор $\tilde{g}_{j_0}$ розраховується подібно (2.4.53) на основі

питомих викидів забруднюючих речовин для окремих технологій $\hat{g}_{j_0}^k$ і інтенсивності використання цих технологій $W_k^{(j_0)}$:

$$\tilde{g}_{j_0} = \sum_{k=1}^{K_{j0}} \tilde{g}_{j_0}^k W_k^{(j_0)} . \qquad (2.4.54)$$

Позначимо $\tilde{G}$ вектор гранично допустимих викидів забруднюючих речовин; матрицю, створену векторами-стовпцями $\tilde{g}_j$, надалі позначимо $\tilde{g}$.

Крім ресурсів, що використовуються в процесі виробництва, у моделі враховуються ресурси, що використовуються для зміни структури галузевих витрат. Допускається, що такі ресурси витрачаються тільки на зменшення величини коефіцієнтів витрат продукції a_{ij}, $\bar{a}_{ij_0}^k$ і витрат ресурсів g_j, $\bar{g}_{j_0}^k$. При цьому залежність між зменшенням вищевказаних показників і витратами ресурсів буде лінійною, з коефіцієнтами пропорційності $b_{vij}, \tilde{b}_{vik}, \bar{b}_{vj}, \bar{b}_{vk}$, де v – номер ресурсу. Позначимо B_v кількість ресурсу виду v.

Задача полягає у визначенні змін коефіцієнтів прямих витрат Δa_{ij}, $i = \overline{1,n}$, $j = \overline{1,n}$, $j \neq j_0$, величини $\Delta\bar{a}_{ij_0}^k$ питомих витрат для окремих технологій, що складають галузь j_0, інтенсивності використання технологій $\Delta W_k^{(j_0)}$, $k = \overline{1,K}_{j0}$, частки кінцевих доходів у ціні продукції Δq_j і питомих витрат ресурсів Δg_j, $\Delta\bar{g}_{j_0}^k$, $j = \overline{1,n}$, $j \neq j_0$, $k = \overline{1,K}_{j0}$, що задовольняють такі умови:

1. Рівності, що визначають величини Δa_{ij_0} і Δg_{j_0} і випливають з (2.4.50) і (2.4.53):

$$\Delta a_{ij_0} = \sum_{k=1}^{K_{j0}} \left(\left(\bar{a}_{ij_0}^k + \Delta\bar{a}_{ij_0}^k \right) \left(W_k^{(j_0)} + \Delta W_k^{(j_0)} \right) - \bar{a}_{ij_0}^k W_k^{(j_0)} \right),$$

$$\Delta g_{ij_0} = \sum_{k=1}^{K_{j0}} \left(\left(\bar{g}_{ij_0}^k + \Delta \bar{g}_{ij_0}^k \right) \left(W_k^{(j_0)} + \Delta W_k^{(j_0)} \right) - \bar{g}_{ij_0}^k W_k^{(j_0)} \right), \quad (2.4.55)$$

$$\tilde{g}_{j_0} = \sum_{k=1}^{K_{j0}} \Delta \hat{g}_{j_0}^k \left(W_k^{(j_0)} + \Delta W_k^{(j_0)} \right), \ i = \overline{1,n.}$$

2. Обмеження на діапазон зміни змінні моделі, що визначаються з технологічної точки зору:

$$\begin{gathered} \underline{\gamma}_{ij} \le \Delta a_{ij} \le \bar{\gamma}_{ij}, \ \ \underline{\delta}_j \le \Delta g_j \le \bar{\delta}_j, \ \ i, j = \overline{1,n}, \\ \underline{W}_k^{(j_0)} \le \Delta W_k^{(j_0)} \le \bar{W}_k^{(j_0)}, \ \ k = \overline{1, K_{j_0}}, \\ \tilde{\gamma}_{ik} \le \Delta \bar{a}_{ij_0}^k \le \bar{\tilde{\gamma}}_{ik}, \ \ i = \overline{1,n}, \ \ k = \overline{1, K_{j_0}}, \\ \underline{\tilde{\delta}}_{ik} \le \Delta \bar{g}_{j_0}^k \le \bar{\tilde{\delta}}_k, \ \ k = \overline{1, K_{j_0}}, \\ 0 \le q_j + \Delta q_j \le 1, \ \ j = \overline{1,n}, \\ 0 \le a_{ij} + \Delta a_{ij} \le 1, \ \ i, j = \overline{1,n}, \end{gathered} \quad (2.4.56)$$

де $\underline{\gamma}_{ij}, \bar{\gamma}_{ij}, \underline{\delta}_j, \bar{\delta}_j, \underline{W}_k^{(j_0)}, \bar{W}_k^{(j_0)}, \tilde{\gamma}_{ik}, \bar{\tilde{\gamma}}_{ik}, \underline{\tilde{\delta}}_{ik}, \bar{\tilde{\delta}}_k$ – деякі наперед задані величини.

3. Обмеження, що випливають зі змісту поняття «частка доданої вартості в ціні продукції»:

$$0 \le a_{jj} + \Delta a_{jj} + l_j \left(q_j + \Delta q_j \right) + d_j \le 1, \qquad j = \overline{1,n}. \quad (2.4.57)$$

4. Обмеження, які унеможливлюють посилення інфляції витрат:

$$\sum_{i=1, i \ne j}^{n} \frac{a_{ij} + \Delta a_{ij}}{1 - \left(a_{jj} + \Delta a_{jj} \right) - \left(l_j \left(q_j + \Delta q_j \right) + d_j \right)} \le \beta, \quad j = \overline{1,n}, \quad (2.4.58)$$

де $\beta < 1$ – деяка наперед задана величина.

5. Ресурсні обмеження для структурно-технологічних змін:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} b_{vij} \max\left(0, -\Delta a_{ij}\right) + \sum_{i=1}^{n} \sum_{k=1}^{K_{j0}} \tilde{b}_{vik} \max\left(0, -\Delta \bar{a}_{ij_0}^k\right) +$$

$$+\sum_{j=1,\, j\neq j_0}^{n} \bar{b}_{\nu j} \max\left(0, -\Delta g_j\right) + \sum_{k=1}^{K_{j_0}} \bar{b}_{\nu k} \max\left(0, -\Delta g_{j_0}^{k}\right) \le B_\nu \,. \quad (2.4.59)$$

6. Рівність, яка визначає прибуток кінцевих споживачів D після проведення структурно-технологічних змін:

$$D = \frac{\left(q+\Delta q\right)^T \left(E - \left(A+\Delta A\right)\right)^{-1} h}{1 - \left(q+\Delta q\right)^T \left(E - \left(A+\Delta A\right)\right)^{-1} \alpha}, \quad (2.4.60)$$

де $A = \left\{a_{ij}\right\}$ – матриця Леонтьєва, $i, j = \overline{1,n}$,

$\Delta A = \left\{\Delta a_{ij}\right\}$ – матриця змін до A, $i, j = \overline{1,n}$,

E – одинична матриця, $q = \left(q_1, ..., q_n\right)$,

$\Delta q = \left(\Delta q_1, ..., \Delta q_n\right)$, $h = \left(h_1, ..., h_n\right)$, $\alpha = \left(\alpha_1, ..., \alpha_n\right)$,

T – операція транспонування.

7. Обмеження на ресурси, що використовуються в процесі виробництва:

$$\left(g+\Delta g\right)^T \left(E - \left(A+\Delta A\right)\right)^{-1} \left(\alpha D + h\right) \le G, \quad (2.4.61)$$

де Δg – матриця з векторів-стовпців Δg_j, $j = \overline{1,n}$.

8. Обмеження на викиди забруднюючих речовин:

$$\tilde{g}^T \left(E - \left(A+\Delta A\right)\right)^{-1} \left(\alpha D + h\right) \le \tilde{G}. \quad (2.4.62)$$

9. Сума нових значень інтенсивностей використання технологій повинна дорівнювати 1 (сукупні зміни цих інтенсивностей дорівнюють нулеві):

$$\sum_{k=1}^{K_{j_0}} \Delta W_k^{(j_0)} = 0. \quad (2.4.63)$$

Серед усіх допустимих розв'язків, що задовольняють співвідношенням (2.4.55) – (2.4.63), варто вибрати такі, що:

а) максимізують кінцеві доходи споживачів:

$$F_1 = D \to \max; \quad (2.4.64)$$

б) максимізують значення мультиплікатора «приріст доходів–приріст виробництва»:

$$F_2 = \left(g + \Delta g\right)^T \left(E - \left(A + \Delta A\right)\right)^{-1} \alpha^{(1)} \to \max, \qquad (2.4.65)$$

де $\alpha^{(1)} = (\alpha_1^{(1)}, ..., \alpha_n^{(1)})$ – деякий заздалегідь визначений числовий вектор;

в) мінімізують витрати особливо дефіцитного ресурсу з виробничою метою:

$$F_3 = \left(g^{(1)} + \Delta g^{(1)}\right)^T \left(E - \left(A + \Delta A\right)\right)^{-1} \left(\alpha D + h\right) \to \min, \quad (2.4.66)$$

де $g^{(1)}$ – рядок матриці g, що містить коефіцієнти питомих витрат цього ресурсу, $\Delta g^{(1)}$ – відповідний рядок матриці Δg.

Відзначимо, що отримана модель (2.4.55) – (2.4.66) буде досить складною оптимізаційною задачею. Цільові функції F_2 і F_3, а також обмеження (2.4.55), (2.4.58) – (2.4.63) є нелінійними. Частина з цих обмежень є рівностями, що призводить до неопуклості множини допустимих розв'язків і додатково ускладнює задачу; функції F_2 і F_3 також неопуклі. Тому необхідна розробка спеціальних алгоритмів для розв'язування такої задачі.

Оптимізаційна задача (2.4.55) – (2.4.66) належить до того ж класу, що і задачі (2.4.9), (2.4.11) – (2.4.16) і (2.4.10) – (2.4.16). Тому розробка вищезгаданих алгоритмів може здійснюватися на основі підходів, викладених у попередніх підрозділах.

2.4.10 Висновки

Розглянуті міжгалузеві моделі планування структурно-технологічних змін розвивають підхід, що пропонувався В.М. Глушковим ще в 70-і роки в публікаціях по системі ДИСПЛАН [14]. Для моделей міжгалузевого балансу це знайшло відображення у виборі цільової функції і побудові системи обмежень, що привели до досить складних оптимізаційних задач з неопуклими цільовими

функціями і нелінійними обмеженнями. Для їхнього розв’язання можна використовувати сучасний арсенал методів недиференційовної оптимізації і методів локального пошуку в сполученні зі спеціальними схемами методу гілок та границь.

Відзначимо, що розглянуті моделі призначені, головним чином, для визначення основних напрямків зміни технологічних коефіцієнтів. Тому вони розраховані на порівняно невеликі розміри міжгалузевих моделей. Розробка міжгалузевих моделей для деталізованої номенклатури галузей (порядку 100 найменувань) вимагатиме удосконалювання подібних моделей для відображення зв'язків номенклатури однієї галузі з номенклатурою іншої. Їх можна відобразити за допомогою обмежень, що описують зв'язки між блоками технологічної матриці.

Список літератури

1. Сергиенко И.В., Михалевич М.В., Стецюк П.И., Кошлай Л.Б. Межотраслевая модель планирования структурно-технологических изменений. Кибернетика и системный анализ. 1998. № 3. С. 3–17.
2. Сергиенко И.В., Михалевич М.В., Стецюк П.И., Кошлай Л.Б. Модели и информационные технологии для поддержки принятия решений при проведении структурно-технологических преобразований. Кибернетика и системный анализ. 2009. № 2. С. 26–49.
3. Sergienko, I.V., Mikhalevich, M.V., Koshlai, L.B., Optimization Models in a Transitional Economy. Series Title: Optimization and its Application. NY: Springer, 2014. 334 p.
4. Михалевич В.С., Михалевич М.В. Динамические макромодели процессов ценообразования в переходной экономике. Кибернетика и системный анализ. 1995. № 3. С. 28–49.
5. Шор Н.З. Методы минимизации недифференцируемых функций и их приложения. Киев: Наук. думка, 1979. 199 с.

6. Shor N.Z. Nondifferentiable Optimization and Polynomial Problems. Dordrecht: Kluwer, 1998. 394 p.
7. Шор Н.З., Стецюк П.И. Использование модификации r-алгоритма для нахождения глобального минимума полиномиальных функций. Кибернетика и системный анализ. 1997. № 4. С. 28–49.
8. Шор Н.З., Стеценко С.И. Квадратичные экстремальные задачи и недифференцируемая оптимизация. Киев: Наукова думка, 1989. 208 с.
9. Форсайт Дж., Малькольм М., Моулер К. Машинные методы математических вычислений. М.: Мир, 1980. 279 с.
10. Субградієнтні алгоритми та задачі на комбінаторних конфігураціях / Стецюк П. І., Донець Г. П., Ненахов Е. І. та ін.; за загал. ред. П. І. Стецюка. Київ: «Унів. вид-во ПУЛЬСАРИ», 2019. 235 с.
11. Стецюк П.И. Об одном методе для нахождения допустимой точки выпуклого неравенства. Теорія оптимальних рішень. Київ: Ін-т кібернетики ім. В.М. Глушкова НАН України, 2000. С. 3–10.
12. Институциональные и технологические изменения в странах с рыночной и переходной экономикой / Стецюк П.И., Бортис Г., Эмменегер Ж.-Ф. и др.; под общей ред. д.ф.-м.н. Стецюка П.И. К.: ВД "Києво-Могилянська академія", 2015. 336 с.
13. Пономаренко О.І., Перестюк М.О., Бурим В.М. Сучасний економічний аналіз. Ч.2. Макроекономіка. Київ: Вища школа, 2004. 208 с.
14. Глушков В.М. Макроэкономические модели и принципы построения ОГАС. М.: Статистика, 1975. 159 с.

Розділ 3. r-АЛГОРИТМ В ПРИКЛАДНИХ ЗАДАЧАХ ОПТИМІЗАЦІЇ

3.1 ОПТИМІЗАЦІЙНІ ЗАДАЧІ ДЛЯ МАКСИМАЛЬНОГО K-ПЛЕКСА

П. І. Стецюк, О. М. Хом'як, А. А. Супрун, Є. А. Блохін

Анотація. Досліджуються дві оптимізаційні задачі, які призначені для знаходження максимального *k*-плекса у неорієнтованому графі. Перша задача є квадратичною оптимізаційною задачею, для неї описано застосування техніки лагранжевих двоїстих оцінок. Друга задача є задачею булевого лінійного програмування, для неї описано алгоритм знаходження усіх розв'язків.

Abstract. Two optimization problems are studied, which are designed to find the maximum *k*-plex in an undirected graph. The first problem is a quadratic optimization problem, for which the application of the Lagrangian dual estimation technique is described. The second problem is a Boolean linear programming problem, for which the algorithm for finding all solutions is described.

3.1.1 Вступ

Поняття k-плекса для неорієнтованого графа введено в [1] (k – деяке натуральне число). Якщо $k=1$, то k-плекс збігається з клікою (повним підграфом) графа. При $k>1$ k-плекс є ослабленням поняття кліки графа і вимагає слабкіші вимоги на включення вершини в k-плекс, ніж вимоги на включення вершини в кліку. Ці поняття широко використовуються в соціології для виявлення та дослідження окремих підгруп населення, при кластеризації даних, для оптимізації інформаційних потоків у мережах тощо (див., напр., [2–5]). Слід

зауважити, що як правило, оптимізаційні задачі пошуку максимальних клік та k-плексів є NP-складними.

У підрозділі описано дослідження двох оптимізаційних задач, призначених для знаходження максимального k-плекса у неорієнтованому графі. Перша задача є квадратичною оптимізаційною задачею, для неї описано застосування техніки лагранжевих двоїстих оцінок [6, 7]. Друга задача є задачею булевого лінійного програмування, для неї описано алгоритм знаходження усіх розв'язків [8].

Послідовність викладення матеріалу буде такою: у параграфі 3.1.2 наведено загальні відомості про k-плекс, у параграфі 3.1.3 описано множину допустимих розв'язків для k-плекса графа G за допомогою системи квадратичних обмежень. У параграфі 3.1.4 розглянуто квадратичну задачу знаходження максимального k-плекса та проаналізовано її зв'язок із булевою лінійною задачею. Там же розглянуто сімейства функціонально надлишкових обмежень для уточнення лагранжевих двоїстих оцінок у квадратичній задачі, що базуються на використанні обмежень булевої лінійної задачі.

У параграфі 3.1.5 проаналізовано зв'язок формулювання квадратичної задачі для 1-плекса з відомим формулюванням квадратичної задачі для пошуку максимальної кліки графа. У параграфі 3.1.6 розглянуто лагранжеві двоїсті оцінки для ряду квадратичних задач та показано, що їх можна покращити за допомогою додавання функціонально надлишкових обмежень [9]. Для обчислення лагранжевих двоїстих оцінок використана програма DSQTPr [10]. У параграфі 3.1.7 наведено алгоритм знаходження усіх розв'язків задачі про k-плекс за допомогою пакету GLPK [11].

3.1.2 Загальні відомості про k-плекс

Нехай $G=(V,E)$ – неорієнтований граф із множиною вершин $V=\{1,...,n\}$ та множиною ребер E. Ребро графа G, що зв'язує вершини $i\in V$ та $j\in V$, умовимося позначати $(i,j)\in E$. Для графа G

будемо використовувати також іншу форму його представлення: $G=(V,\Gamma)$, де $\Gamma=\{\Gamma(i),\ i=\overline{1,n}\}$, а $\Gamma(i)$ – кінцеві вершини тих дуг, у яких початковою вершиною є вершина i. Кількість ребер графа G в обох представленнях зв'язані співвідношенням: $|E|=\frac{1}{2}\sum_{i\in V}|\Gamma(i)|$. Комплементарний до G граф будемо позначати $\overline{G}=(V,\overline{E})$, і $\overline{G}=(V,\overline{\Gamma})$, де $(i,j)\in\overline{E}$ і $\overline{\Gamma}=\{\overline{\Gamma}(i),\ i=\overline{1,n}\}$.

Означення 3.1.1. Підмножина вершин S із V називається k-плексом графа G, якщо ступінь кожної вершини в індукованому підграфі $G[S]$ (підграфі, породженому підмножиною S) є не менша, ніж $|S|-k$. Тобто $S\subset V$ є k-плексом, якщо виконується така умова:

$$\deg_{G[S]}(i)=|\Gamma(i)\cap S|\geq|S|-k \quad \forall i\in S.$$

k-Плекс є максимальним по включенню (maximal), якщо він не міститься ні в якому іншому k-плексі. Найбільший з максимальних по включенню k-плексів називається максимальним (maximum), його розмір називається k-плексним числом графа G та позначається $\rho_k(G)$ [1]. Очевидно, що 1-плекс є клікою графа G, тому що ступінь кожної вершини в індукованому підграфі $G[S]$ не менша, ніж $|S|-1$, а це означає, що кожна з вершин у підграфі $G[S]$ зв'язана з усіма іншими вершинами, тобто підграф $G[S]$ є повним підграфом (клікою) графа G. У даному випадку $\rho_1(G)=\omega(G)$, де $\omega(G)$ – клікове число графа G, що відповідає розміру максимальної кліки в графі G.

Поняття $co-k$-плекса графа G, також введене в [1], є узагальненням поняття незалежної множини вершин графа G. При $k=1$ $co-k$-плекс збігається з незалежною множиною вершин графа. При $k>1$ $co-k$-плекс є ослабленням поняття незалежної множини вершин графа (відомої також як внутрішньо стійка множина).

Означення 3.1.2. Підмножина вершин S з V називається $co\text{-}k$-плексом графа G, якщо виконується така умова:

$$\deg_{G[S]}(i)=\left|\Gamma(i)\cap S\right|\le k-1 \quad \forall i\in S .$$

Отже, $S\subset V$ є $co\text{-}k$-плексом, якщо ступінь кожної вершини в індукованому підграфі $G[S]$ є не більшою, ніж $k-1$. Очевидно, що CO-1-плекс є незалежною множиною вершин графа G, оскільки ступінь кожної вершини в індуційованому підграфі $G[S]$ дорівнює нулю, а це означає, що кожна з вершин у підграфі $G[S]$ не зв'язана з жодною з інших вершин підграфа $G[S]$. Відзначимо, що $co\text{-}k$-плекс і k-плекс для графа G знаходяться в такому ж зв'язку як кліка графа G і незалежна множина вершин графа G. Тому підмножина S є $co\text{-}k$-плексом графа G тоді й тільки тоді, коли S є k-плексом для комплементарного графа G.

3.1.3 Квадратичні обмеження для *k*-плекса

Нехай вершині $i\in V$ $(i=1,2,...)$ відповідає булева змінна $x_i\in\{0,1\}$ така, що

$$x_i=\begin{cases}1, & якщо\ \ i\in S,\\ 0, & якщо\ \ i\in V\setminus S.\end{cases}$$

Булеві змінні x_i, $i=\overline{1,n}$ будуть описуватися за допомогою квадратичних обмежень-рівностей

$$x_i^2-x_i=0 \quad \forall i\in V . \qquad (3.1.1)$$

Побудуємо такі квадратичні обмеження, щоб підмножина S була k-плексом. Ці обмеження повинні задавати вимоги на те, щоб ступінь кожної вершини $i\in S$ у підграфі $G[S]$ була не меншою за $|S|-k$, інакше кажучи, щоб у підграфі $G[S]$ кількість дуг, що виходять із кожної вершини $i\in S$ була не меншою за $|S|-k$.

Нехай вершина i належить підмножині S, тобто $x_i = 1$. Позначимо $N_e(i)$ ступінь вершини i у підграфі $G[S]$, тобто кількість дуг, що виходять з вершини $i \in S$. Тоді в підграфі $G[S]$ ступені вершин із підмножини S задаються за допомогою сімейства співвідношень:

$$N_e(i) = \sum_{j \in \Gamma(i)} x_j, \quad \forall i \in S. \tag{3.1.2}$$

З рівняння $|S| = \sum_{j \in V} x_j$ та умови, що множина S є k-плексом, одержуємо нерівності

$$N_e(i) \geq |S| - k = \sum_{j \in \Gamma(i)} x_j - k, \quad \forall i \in S. \tag{3.1.3}$$

Із співвідношень (3.1.2) та (3.1.3) одержуємо сімейство нерівностей

$$\sum_{j \in \Gamma(i)} x_j \geq \sum_{j \in V} x_j - k, \quad \forall i \in S, \tag{3.1.4}$$

при виконанні яких всі ті вершини $i \in V$, для яких $x_i = 1$, будуть утворювати k-плекс.

Однак, нерівності типу (3.1.4) не будуть виконуватися для тих вершин $i \in V$, для яких $x_i = 0$, тобто для всіх $x_i \in V \setminus S$. Для того, щоб одержати квадратичні нерівності, що будуть справедливими і для змінних $x_i = 0$, досить обидві частини нерівності вигляду (3.1.4), що відповідає вершині i, помножити на змінну x_i. З огляду на невід'ємність змінної x_i знак нерівності після множення не зміниться, і в результаті одержимо такі нерівності:

$$x_i \left(\sum_{j \in \Gamma(i)} x_j \right) \geq x_i \left(\sum_{j \in V} x_j - k \right), \quad \forall i \in V,$$

які можна переписати у вигляді

$$\sum_{j \in \Gamma(i)} x_i x_j \geq \sum_{j \in V} x_i x_j - k x_i, \quad \forall i \in V. \tag{3.1.5}$$

Квадратичні нерівності (3.1.5) разом із обмеженнями (3.1.1) повністю описують умови, за яких вершини i належать k-плексу. Дійсно, нерівності (3.1.5) будуть справедливі для тих вершин i, для яких $x_i = 1$, тому що вони переходять в обмеження (3.1.4). Нерівності (3.1.5) будуть справедливі також для усіх вершин i, для яких $x_i = 0$, тому що вони переходять у тривіальну нерівність $0 \geq 0$.

Зрозуміло, що за допомогою обмежень у вигляді рівностей (3.1.1) та у вигляді нерівностей (3.1.5) можна описати допустимі булеві розв'язки, що відповідають k-плексу. При цьому зміст обмежень (3.1.5) буде пов'язаний із інтерпретацією ступеня вершини, як цього вимагає поняття k-плекса, тобто ступінь вершини $i \in S$ більше або дорівнює $|S| - k$. Дійсно, права частина обмеження (3.1.5) для вершини i, що належить k-плексу, вказує кількість ребер, що виходять з i-ї вершини, з урахуванням того, що $x_i x_j = 1$ лише тоді, коли обидві змінні x_i та x_j дорівнюють одиниці.

Нерівність (3.1.5) можна спростити. Зауважимо, що

$$\sum_{j \in V} x_j = \sum_{j \in \Gamma(i)} x_j + \sum_{j \in \overline{\Gamma}(i)} x_j + x_i .$$

Тоді нерівності (3.1.5) можна переписати так:

$$\sum_{j \in \Gamma(i)} x_i x_j \geq \left(\sum_{j \in \Gamma(i)} x_i x_j + \sum_{j \in \overline{\Gamma}(i)} x_i x_j + x_i^2 \right) - k x_i, \quad \forall i \in V ,$$

звідки

$$\sum_{j \in \overline{\Gamma}(i)} x_i x_j \leq k x_i - x_i^2, \quad \forall i \in V .$$

З урахуванням того, що $x_i = x_i^2$ (див. формулу (3.1.1)), останні нерівності можна переписати як

$$\sum_{j \in \overline{\Gamma}(i)} x_i x_j \leq (k-1) x_i, \quad \forall i \in V . \qquad (3.1.6)$$

Нерівності (3.1.6) разом із рівностями (3.1.1) ми покладемо в основу квадратичної моделі для знаходження максимального k-плекса графа G.

Зауважимо, що квадратичним нерівностям (3.1.6) можна надати інший зміст, ніж нерівностям (3.1.5). Нерівності (3.1.6) пов'язані з комплементарним графом $\overline{G}$ і описують таку підмножину вершин S, що ступінь вершини в індуційованому цією підмножиною підграфі $\overline{G}[S]$ була не більшою за $(k-1)$. Дійсно, для тих вершин $i \in V$, для яких $x_i = 1$, нерівності (3.1.6) рівносильні таким нерівностям

$$\sum_{j \in \overline{\Gamma}(i)} x_j \le (k-1), \quad \forall i \in S,$$

а для тих вершин i, для яких $x_i = 0$, вони рівносильні тривіальним нерівностям $0 \le 0$. Тому опис підмножини S за допомогою нерівностей (3.1.6) та рівностей (3.1.1) логічно інтерпретувати як опис co-k-плекса для комплементарного графа $\overline{G}$.

3.1.4 Квадратична булева задача для $\rho_k(G)$

Враховуючи, що обмеження (3.1.1) та (3.1.6) описують множину допустимих варіантів утворення k-плекса, для знаходження максимального k-плекса графа G оптимізаційну квадратичну задачу можна сформулювати в такій формі:

$$\rho_k(G) = \max_x \sum_{i \in V} x_i \tag{3.1.7}$$

за обмежень

$$\sum_{j \in \overline{\Gamma}(i)} x_i x_j \le (k-1) x_i, \quad \forall i \in V, \tag{3.1.8}$$

$$x_i^2 - x_i = 0, \quad \forall i \in V. \tag{3.1.9}$$

Зрозуміло, що задачу (3.1.7) – (3.1.9) можна інтерпретувати як задачу знаходження максимального co-k-плекса графа $\overline{G}$. Із (3.1.7) – (3.1.9) легко одержати формулювання квадратичної оптимізаційної задачі для

$co\text{-}k$-плекса графа $\overline{G}$. Для цього досить у сумі лівої частини обмеження (3.1.8) замість підсумовування по $j \in \overline{\Gamma}(i)$ використовувати підсумовування по $j \in \Gamma(i)$. Інакше кажучи, якщо обмеження (3.1.8) замінити на

$$\sum_{j \in \Gamma(i)} x_i x_j \le (k-1) x_i, \quad \forall i \in V, \tag{3.1.10}$$

то ми одержимо формулювання квадратичної оптимізаційної задачі знаходження максимального $co\text{-}k$-плекса графа $\overline{G}$.

Формулювання задачі (3.1.7) – (3.1.9) можна одержати із задачі булевого лінійного програмування, запропонованої в [2]:

$$\rho_k(G) = \max_x \sum_{i \in V} x_i \tag{3.1.11}$$

за обмежень

$$\sum_{j \in \overline{\Gamma}(i)} x_j \le (k-1) x_i - \overline{d}_i (1 - x_i), \quad \forall i \in V, \tag{3.1.12}$$

$$x_i \in \{0,1\}, \quad \forall i \in V, \tag{3.1.13}$$

Де $\overline{d}_i = \left|\overline{\Gamma}(i)\right|$. Лінійні обмеження (3.1.12) побудовані за схемою, подібною тій, що використовувалася у попередньому розділі при переході від обмежень (3.1.4), справедливих для $i \in S$ (тобто коли $x_i = 1$), до обмежень (3.1.5), що є справедливими також для $i \in V \setminus S$ (тобто коли $x_i = 0$). Однак, правило для того, щоб обмеження (3.1.12) виконувалися для будь-яких $i \in V \setminus S$, тут буде іншим. Так, коли $x_i = 1$, тобто обмеження (3.1.12) виконуються як нерівності

$$\sum_{j \in \overline{\Gamma}(i)} x_j \le (k-1), \quad \forall i \in S,$$

які повинні бути справедливими для $co\text{-}k$-плекса графа $\overline{G}$, що збігається з k-плексом графа G. Коли $x_i = 0$, то обмеження (3.1.12) переходять у нерівності

$$\sum_{j\in\overline{\Gamma}(i)} x_j \le \overline{d}_i = \left|\overline{\Gamma}(i)\right|, \quad \forall i \in V \setminus S ,$$

які є справедливими, бо в якості верхньої границі на ступені вершин, що не входять у $co\text{-}k$-плекс графа $\overline{G}$, використовується максимальна можлива кількість ребер, що виходять з кожної з вершин графа $\overline{G}$. Ті з цих обмежень, де не всі змінні під знаком суми дорівнюють одиниці, будуть надлишковими.

Із задачі лінійного булевого програмування (3.1.11) – (3.1.13) легко одержати квадратичну задачу (3.1.7) – (3.1.9). Для цього слід обмеження (3.1.13) замінити на відповідний нелінійний аналог (3.1.9), а обмеження, яке відноситься до i-ї вершини з (3.1.12), помножити на змінну x_i. В силу невід'ємності змінних x_i, $i = 1, 2, \ldots$, знаки нерівностей при множенні не зміняться, і в результаті одержимо

$$\sum_{j\in\overline{\Gamma}(i)} x_i x_j \le (k-1)x_i^2 + \overline{d}_i (1-x_i) x_i, \quad \forall i \in V ,$$

звідки, з урахуванням того, що $(1-x_i)x_i = x_i - x_i^2 = 0$ для всіх $i \in V$, приходимо до обмежень (3.1.8).

Зрозуміло, що кожна з задач (3.1.7) – (3.1.9) та (3.1.11) – (3.1.13) має свої переваги та свої недоліки. Так, наприклад, найсуттєвіша перевага квадратичної задачі над лінійною булевою полягає в тому, що незначним удосконаленням задачі (3.1.7) – (3.1.9) можна сформулювати квадратичні оптимізаційні задачі знаходження максимальних за розміром підмножин графа із сильнішими властивостями, ніж k-плекс або $co\text{-}k$-плекс, – достатньо лише посилити вимогу на включення вершин у ці підмножини. Так, наприклад, умовимося під "строгим" k-плексом графа G розуміти підмножину його вершин, для яких ступінь вершини дорівнює $|S| - k$. Аналогічно введемо поняття строгого $co\text{-}k$-плекса: в ньому ступінь вершини дорівнює рівно $k-1$. Тоді для того, щоб сформулювати квадратичні задачі знаходження максимальних із цих підмножин, досить в обмеженнях (3.1.8) та (3.1.10) замість нерівностей

використовувати рівності. Інакше кажучи, для знаходження "строгого" k-плекса графа G обмеження (3.1.8) слід замінити на обмеження

$$\sum_{j\in\overline{\Gamma}(i)} x_i x_j \le (k-1)x_i, \quad \forall i \in V,$$

а для знаходження строгого $co\text{-}k$-плекса графа G обмеження (3.1.10) слід замінити на такі:

$$\sum_{j\in\Gamma(i)} x_i x_j \le (k-1)x_i, \quad \forall i \in V.$$

До переваги лінійної булевої задачі над квадратичною можна віднести той факт, що для задачі (3.1.11) – (3.1.13) легко підраховувати верхні оцінки для $\rho_k(G)$ за допомогою релаксації обмеження (3.1.13). У ряді випадків, наприклад, коли $\rho_k(G)$ буде більше $n/3$, ці оцінки, як правило, можуть виявитися ефективними оцінками зверху для $\rho_k(G)$. У результаті для деяких спеціальних графів на базі методу гілок та границь можна реалізувати швидкі алгоритми для знаходження $\rho_k(G)$.

Знаходження верхніх оцінок для квадратичної задачі (3.1.7) – (3.1.9) є більш трудомістким, ніж для релаксованої задачі (3.1.11) – (3.1.13). Так, наприклад, якщо в такій якості використовувати лагранжеві двоїсті оцінки [9, 12], то знаходження таких оцінок за допомогою методів недиференційованої оптимізації вимагатиме більше часу, ніж у випадку задач лінійного програмування. Однак, для ряду графів лагранжеві двоїсті оцінки можуть виявитися значно точнішими верхніми оцінками, ніж лінійні оцінки. Більш того, існує резерв для уточнення лагранжевих двоїстих оцінок задачі (3.1.7) – (3.1.9): це введення функціонально надлишкових обмежень [9].

Лінійні обмеження (3.1.12) можна використовувати для побудови функціонально надлишкових обмежень з метою покращити точність лагранжевих двоїстих оцінок у багатоекстремальних квадратичних задачах (3.1.7) – (3.1.9). Якщо скористатися схемою, що використовувалася Н.З. Шором для задачі про максимальну незалежну

множину вершин графа [9, ст. 250], то при цьому до задачі (3.1.7) – (3.1.9) додаються два види функціонально надлишкових обмежень. Обмеження першого виду одержано домноженням кожного з лінійних обмежень у (3.1.12) на ті змінні x_l, які не входять у це обмеження, тобто, додаються $n(n-1)$ надлишкових обмежень вигляду

$$\sum_{j\in\bar{\Gamma}(i)} x_i x_j \le (k-1)x_l x_i + \bar{d}_i (1-x_i)x_l, \quad \forall i,l \in V,\ i \ne l\ . \qquad (3.1.14)$$

Функціонально надлишкові обмеження другого типу можна одержати з лінійних обмежень (3.1.12), помноживши на $1-x_l$, $l=1,2,\ldots$. Тут уже можна використовувати $i=l$, бо вони дають нові квадратичні обмеження у формі нерівностей. У результаті маємо n^2 обмежень

$$\sum_{j\in\bar{\Gamma}(i)} x_l x_j - \sum_{j\in\bar{\Gamma}(i)} x_j \le (k-1)x_l x_i + \bar{d}_i (1-x_i)x_l, \quad \forall i,l \in V\ . \qquad (3.1.15)$$

Зауважимо, що використання надлишкових обмежень (3.1.14), (3.1.15) значно збільшує розміри квадратичної задачі. Однак, якщо до попередньої задачі додавати тільки невелику кількість тих надлишкових обмежень, які уточнюють двоїсту оцінку для наступної задачі, то тоді загальна кількість обмежень у кінцевій квадратичній задачі буде невеликою. За такою ж схемою можна використовувати інші види функціонально надлишкових обмежень для булевих задач, які розглядалися у роботах [13, 14].

3.1.5 Максимальний 1-плекс та максимальна кліка

Оскільки для графа G 1-плекс збігається з клікою, природно, що знаходження максимального 1-плекса графа G у задачі (3.1.7) – (3.1.9) відповідає знаходженню максимальної кліки графа G. Якщо $k=1$, то права частина нерівностей (3.1.8) дорівнює нулю, і задача (3.1.7) – (3.1.9) переходить у таку квадратичну оптимізаційну задачу:

$$\rho_1(G) = \omega(G) = \max \sum_{i\in V} x_i \qquad (3.1.16)$$

за обмежень

$$\sum_{j\in\overline{\Gamma}(i)} x_i x_j \le 0, \quad \forall i \in V, \tag{3.1.17}$$

$$x_i^2 - x_i = 0, \quad \forall i \in V. \tag{3.1.18}$$

Задачу (3.1.16) – (3.1.18) можна розглядати як одне з можливих квадратичних формулювань задачі про знаходження максимальної кліки графа G. Задачі (3.1.16) – (3.1.18) відповідає рівно n квадратичних обмежень нерівностей (3.1.17).

Більш відомим квадратичним формулюванням задачі для максимальної кліки графа G є квадратична оптимізаційна задача [2]:

$$\omega(G) = \max \sum_{i\in V} x_i \tag{3.1.19}$$

за обмежень

$$x_i x_j = 0, \quad \forall (i, j) \in \overline{E}, \tag{3.1.20}$$

$$x_i^2 - x_i = 0, \quad \forall i \in V. \tag{3.1.21}$$

Задачу (3.1.19) – (3.1.21) можна інтерпретувати як очевидний наслідок задачі (3.1.16) – (3.1.18), тому що всі змінні x_i, $i = \overline{1,n}$ є невід'ємними. Дійсно, згідно з квадратичними обмеженнями (3.1.17), сума невід'ємних добутків пар змінних повинна бути не більше нуля, а це еквівалентно (за умови невід'ємності) системі квадратичних рівностей, де кожен окремий добуток двох змінних, котрі входять в обмеження (3.1.17), дорівнює нулю. Це й становить зміст обмежень (3.1.20).

Яка з квадратичних моделей є кращою для знаходження максимальної кліки графа G? Формально задача (3.1.16) – (3.1.18) має ту перевагу, що вона містить меншу кількість квадратичних обмежень, ніж задача (3.1.19) – (3.1.21). Однак, вона програє в точності лагранжевої двоїстої оцінки.

Нехай $\psi_{\rho_1}^*(G)$ – оптимальна лагранжева оцінка для задачі (3.1.16) – (3.1.18), а $\psi_{\omega}^*(G)$ – оптимальна лагранжева оцінка для задачі (3.1.19) – (3.1.21). Кожна з них буде верхньою оцінкою для $\omega(G)$.

Верхня оцінка $\psi^*_{\rho_1}(G)$ буде, як правило, гіршою за оцінку $\psi^*_{\omega}(G)$. Це можна пояснити тим, що в задачі (3.1.19) – (3.1.21) ми маємо більшу свободу у виборі множників Лагранжа, що відповідають обмеженням (3.1.20): кожному з ребер комплементарного графа $\overline{G}$ буде відповідати свій множник Лагранжа. Для обмежень (3.1.17) таких множників Лагранжа усього n, тобто свій множник Лагранжа буде відповідати кожній з вершин графа G. Крім того, свобода на вибір множників Лагранжа для обмежень (3.1.17) обмежена ще й тим, що для кожного з цих множників потрібно врахувати його невід'ємність. Урахування невід'ємності обмежує можливості вибору множників Лагранжа в задачі (3.1.16) – (3.1.18) у порівнянні з випадком, коли обмеження (3.1.17) замінюються на

$$\sum_{j\in\overline{\Gamma}(i)} x_i x_j = 0, \quad \forall i \in V$$

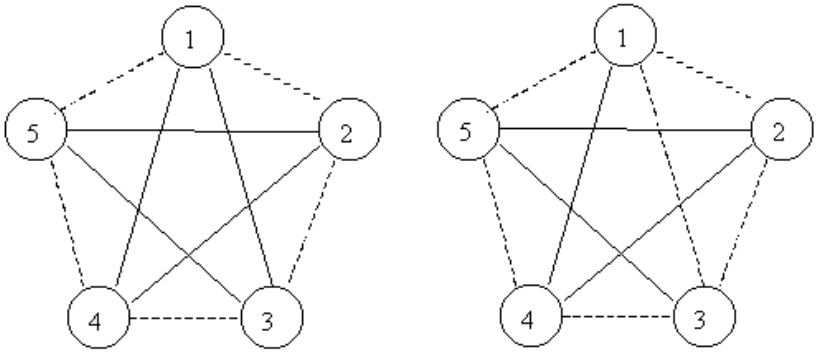
(вони випливають із обмежень (3.1.20), якщо їх згрупувати по вершинах графа G). Зауважимо, що квадратичну задачу з цими обмеженнями замість обмежень (3.1.17) можна розглядати як задачу знаходження максимального «строгого» 1-плекса графа G. Природно, що лагранжева двоїста оцінка для квадратичної задачі, що відповідає максимальному «строгому» 1-плексу графа G, буде в багатьох випадках точнішою за оцінку $\psi^*_{\rho_1}(G)$.

Проілюструємо поводження лагранжевих двоїстих оцінок $\psi^*_{\rho_1}(G)$ та $\psi^*_{\omega}(G)$ на прикладі двох графів G_1 і G_2 (рис. 3.1.1), що складаються усього з п'яти вершин. Тут ребра графів G_1 і G_2 позначено суцільною лінією, а ребра комплементарних графів $\overline{G}_1$ і $\overline{G}_2$ – пунктирною. Для графа G_1 (рис. 3.1.1а)) системи обмежень для задач (3.1.16) – (3.1.18) та (3.1.19) – (3.1.21) будуть такими:

$$x_1x_2 + x_1x_5 \le 0, \quad x_1x_2 = 0,$$
$$x_1x_2 + x_2x_3 \le 0, \quad x_2x_3 = 0,$$
$$x_2x_3 + x_3x_4 \le 0, \quad x_3x_4 = 0,$$

$$x_3x_4 + x_4x_5 \le 0, \quad x_4x_5 = 0,$$
$$x_4x_5 + x_1x_5 \le 0, \quad x_1x_5 = 0.$$

Лагранжеві двоїсті оцінки $\psi^*_{\rho_1}(G)$ та $\psi^*_{\omega}(G)$ є однаковими й дорівнюють $\sqrt{5}$. Вони є неточними оцінками зверху, тому що $\omega(G_1) = 2$. Приклад для графа G_1 демонструє, що лагранжеві двоїсті оцінки $\psi^*_{\rho_1}(G)$ та $\psi^*_{\omega}(G)$ можуть співпадати.



а) б)

Рис. 3.1.1. Приклади графів G_1 та G_2

Це твердження не виконується для графа G_2 на рис. 3.1.1 б), що відрізняється від G_1 тим, що з нього вилучене ребро (1,3). Для графа G_2 системи обмежень задач (3.1.16) – (3.1.18) та (3.1.19) – (3.1.21) будуть такими:

$$x_1x_2 + x_1x_3 + x_1x_5 \le 0, \qquad x_1x_2 = 0,$$
$$x_1x_2 + x_2x_3 \le 0, \qquad x_1x_3 = 0,$$
$$x_1x_3 + x_2x_3 + x_3x_4 \le 0, \qquad x_2x_3 = 0,$$
$$x_3x_4 + x_4x_5 \le 0, \qquad x_3x_4 = 0,$$

$$x_4 x_5 + x_1 x_5 \le 0 \qquad\qquad x_4 x_5 = 0,$$
$$x_1 x_5 = 0.$$

На цей раз оцінки $\psi^*_{\rho_1}(G)$ та $\psi^*_{\omega}(G)$ – різні. Зокрема, оцінка $\psi^*_{\omega}(G) = 2.0$ є точною верхньою оцінкою для $\omega(G_2)$. Оцінка $\psi^*_{\rho_1}(G) = 2.15222$ не є точною і потребує уточнення. Отже, приклад графа G_2 показує, що лагранжева двоїста оцінка для задачі (3.1.19) – (3.1.21) має перевагу над відповідною оцінкою задачі (3.1.16) – (3.1.18).

Якщо для графа G_2 обмеження нерівності замінити на обмеження у формі рівностей, що відповідає знаходженню максимального «строгого» 1-плекса графа G_1, то лагранжева двоїста оцінка стане рівною двом і буде точною верхньою оцінкою для $\omega(G_1)$.

Якщо потрібно знайти більш точну лагранжеву двоїсту оцінку для максимальної кліки графа G за допомогою одної з двох розглянутих квадратичних задач, то краще користуватися квадратичною задачею у формі (3.1.19) – (3.1.21), ніж квадратичною задачею у формі (3.1.16) – (3.1.18), що для 1-плекса випливає з квадратичного формулювання (3.1.7) – (3.1.9). Аналогічна ситуація буде мати місце і для квадратичної задачі [9], що пов'язана із знаходженням максимальної незалежної множини вершин графа.

Однак, ситуація може змінитися, якщо мова йтиме не про знаходження оцінок $\psi^*_{\rho_1}(G)$ та $\psi^*_{\omega}(G)$, а про їх поліпшення за допомогою додавання функціонально надлишкових обмежень. Тоді квадратична задача (3.1.16) – (3.1.18) може мати переваги хоча б тому, що до неї можна додати більшу кількість функціонально надлишкових обмежень. Крім того, коли $k \ge 2$, то для знаходження максимального k-плекса навряд чи можна знайти альтернативу квадратичному формулюванню (3.1.7) – (3.1.9). Тому дослідження точності лагранжевих двоїстих оцінок для задачі (3.1.7) – (3.1.9), що підсилена за рахунок додавання функціонально надлишкових квадратичних

обмежень, становить значний інтерес. У результаті може виявитися, що безпосереднє перенесення таких оцінок на окремий випадок $k=1$ для ряду графів може привести до досить хороших по точності верхніх оцінок для $\omega(G)$.

Для уточнення верхніх оцінок можно використовувати квадратичні нерівності з двох сімейств функціонально надлишкових квадратичних обмежень – це сімейства квадратичних обмеженнь (3.1.14) та (3.1.15), які описано у параграфі 3.1.3. Для графів G_1 та G_2 з рис. 3.1.1 нижче буде показано, що за допомогою функціонально надлишкових обмежень із сімейств (3.1.14) і (3.1.15) лагранжеві двоїсті оцінки можна зробити точними верхніми оцінками для $\rho_2(G_1)$ та $\rho_2(G_2)$.

3.1.6 Уточнення лагранжевих оцінок для максимального 2-плекса

Розглянемо задачу знаходження максимального 2-плекса для графа G_1 з рис. 3.1.1а). Вона формулюється у вигляді такої квадратичної задачі:

$$\rho_2(G_1)=\max_{x\in\mathbb{R}^5}(x_1+x_2+x_3+x_4+x_5) \tag{3.1.22}$$

за обмежень

$$\begin{cases} x_1x_2+x_1x_5\le x_1,\\ x_1x_2+x_2x_3\le x_2,\\ x_2x_3+x_3x_4\le x_3,\\ x_3x_4+x_4x_5\le x_4,\\ x_4x_5+x_1x_5\le x_5, \end{cases} \tag{3.1.23}$$

$$x_i^2-x_i=0,\quad i=\overline{1,5}. \tag{3.1.24}$$

Тут $\rho_2(G_1)=3$. У той же час лагранжева двоїста оцінка для задачі (3.1.22) – (3.1.24) дорівнює 3.618034. Вона є менш точною, ніж лінійна верхня оцінка, що виходить у результаті релаксації задачі булевого

програмування (3.1.11) – (3.1.13) і дорівнює 10/3 = 3.33333. Задачі лінійного програмування відповідають такі лінійні обмеження:

$$\begin{cases} x_1 + x_2 + x_5 \le 2, \\ x_1 + x_2 + x_3 \le 2, \\ x_2 + x_3 + x_4 \le 2, \\ x_3 + x_4 + x_5 \le 2, \\ x_1 + x_4 + x_5 \le 2, \end{cases} \qquad (3.1.25)$$

які випливають з обмежень (3.1.12).

Як лагранжева двоїста оцінка, так і лінійна оцінка є неточними верхніми оцінками. Покращити лагранжеву двоїсту оцінку, що відповідає задачі (3.1.22) – (3.1.24), можна, додаючи до квадратичної задачі (3.1.22) – (3.1.24) функціонально надлишкові обмеження із сімейства (3.1.14). Вони побудовані домноженням лінійних обмежень (3.1.25) на змінні x_i, $i = \overline{1,n}$. Послідовне додавання таких обмежень робить лагранжеву двоїсту оцінку більш точною. При послідовному додаванні функціонально надлишкових квадратичних обмежень із сімейства (3.1.14), вказаних у табл. 3.1.1, спостерігається монотонне зменшення оцінки ψ^*. У результаті додавання чотирьох додаткових обмежень верхня оцінка ψ^* стає точнішою за лінійну. А в результаті додавання всіх п'яти обмежень оцінка ψ^* стає точною верхньою оцінкою для $\rho_2(G_1)$.

Табл. 3.1.1. Покращення верхньої оцінки для задачі (3.1.22) – (3.1.24)

$+n$	Лінійні обмеження	$\times x_i$	Квадратичні обмеження	ψ^*
+1	$x_1 + x_2 + x_5 \le 2$	$\times x_2$	$x_1x_2 + x_2x_5 \le x_2$	3.56878
+2	$x_1 + x_2 + x_3 \le 2$	$\times x_3$	$x_1x_3 + x_2x_3 \le x_3$	3.52727
+3	$x_2 + x_3 + x_4 \le 2$	$\times x_4$	$x_2x_4 + x_2x_4 \le x_4$	3.42929
+4	$x_3 + x_4 + x_5 \le 2$	$\times x_5$	$x_3x_5 + x_4x_5 \le x_5$	3.21298
+5	$x_1 + x_4 + x_5 \le 2$	$\times x_1$	$x_1x_4 + x_1x_5 \le x_1$	3.00000

Схожа, але дещо інша, ситуація має місце і для задачі знаходження максимального 2-плекса для графа G_2 з рис. 3.1.1б). Вона формулюється у вигляді квадратичної задачі

$$\rho_2\left(G_2\right)=\max_{x\in\mathbb{R}^5}\left(x_1+x_2+x_3+x_4+x_5\right) \tag{3.1.26}$$

за обмежень

$$\begin{cases} x_1x_2+x_1x_3+x_1x_5\le x_1,\\ x_1x_2+x_2x_3\le x_2,\\ x_1x_2+x_2x_3+x_3x_4\le x_3,\\ x_3x_4+x_4x_5\le x_4,\\ x_4x_5+x_1x_5\le x_5, \end{cases} \tag{3.1.27}$$

$$x_i^2-x_i=0,\quad i=\overline{1,5}, \tag{3.1.28}$$

і їй відповідає $\rho_2\left(G_2\right)=3$. Лагранжева двоїста оцінка для задачі (3.1.26) – (3.1.28) дорівнює 3.37332 і є точнішою, ніж лінійна верхня оцінка, що дорівнює 3.4. Задачі лінійного програмування відповідають такі лінійні обмеження виду (3.1.12):

$$\begin{cases} 2x_1+x_2+x_3+x_5\le 3,\\ x_1+x_2+x_3\le 2,\\ x_1+x_2+2x_3+x_4\le 3,\\ x_3+x_4+x_5\le 2,\\ x_1+x_4+x_5\le 2. \end{cases} \tag{3.1.29}$$

Враховуючи, що лагранжева двоїста оцінка є точнішою за лінійну, може скластися враження, що точну оцінку ψ^* простіше одержати за допомогою додавання функціонально надлишкових обмежень (3.1.14), тобто тих, котрі побудовані з лінійних обмежень (3.1.29) домноженням на змінні x_i, $i=\overline{1,5}$. Однак, це не так, і максимальна по точності лагранжева двоїста оцінка, яку можна досягти на цьому шляху, дорівнює 3.02316. Лише при додаванні функціонально надлишкових обмежень у вигляді (3.1.15) можна домогтися того, щоб лагранжева двоїста оцінка стала точною. Наприклад, після додавання

до квадратичної задачі (3.1.26) – (3.1.28) функціонально надлишкового обмеження виду (3.1.14)

$$x_1x_1 + x_1x_4 + x_1x_5 - 2x_1 \le 0,$$

яке одержано із п'ятого обмеження (3.1.29) домноженням на x_1, оцінка ψ^* стає рівною 3.21576. Після додавання обмеження виду (3.1.15)

$$-x_1x_3 - x_1x_4 - x_1x_5 + 2x_1 + x_3 + x_4 + x_5 \le 2,$$

яке отримано домноженням четвертого обмеження з (3.1.29) на $(1-x_1)$, оцінка ψ^* стає рівною 3.01571. А після додавання ще й обмежень виду (3.1.15)

$$-2x_1x_2 - x_2^2 - x_2x_3 - x_2x_5 + 2x_1 + 4x_2 + x_3 + x_5 \le 3,$$

які одержано домноженням першого обмеження з (3.1.29) на $(1-x_2)$, оцінка ψ^* стає точною й дорівнює 3.0.

3.1.7 Застосування GLPK для знаходження всіх розв'язків

Задача (3.1.11) – (3.1.13) може мати як один, так і багато розв'язків. Для того, щоб знайти усі її розв'язки, можна використати алгоритм [19], який полягає у послідовному доповненні задачі додатковим лінійним обмеженням, яке відсікає уже знайдені розв'язки. Нехай уже знайдено m максимальних k-плексів S_j, $j=\overline{1,m}$. Додамо до задачі (3.1.11) – (3.1.13) таке сімейство лінійних обмежень:

$$\sum_{i\in V(S_j)} x_i \le \rho_k(G) - 1/2, \quad j=\overline{1,m}. \qquad (3.1.30)$$

Якщо в задачі (3.1.11) – (3.1.13), (3.1.30) значення цільової функції дорівнює значенню цільової функції у задачі (3.1.11) – (3.1.13), то це означає, що ми знайшли новий $(m+1)$ максимальний k-плекс, який не співпадає ні з одним із m існуючих. Якщо в задачі (3.1.11) – (3.1.13), (3.1.30) значення цільової функції є меншим за значення цільової функції у задачі (3.1.11) – (3.1.13), то це означає, що

ми отримали достатню умову того, що інших максимальних k-плексів не існує, окрім тих m, які були знайдені раніше.

Описаний алгоритм для пошуку $n_{sol} <$ **ncalls** розв`язків у задачі (3.1.11) – (3.1.13) реалізований мовою Octave. Задачі лінійного булевого програмування (3.1.11) – (3.1.13) та (3.1.11) – (3.1.13), (3.1.30) розв'язуються за допомогою octave-функції GLPK [11], яка використовує відомий пакет GLPK (GNU Linear Programming Kit) для розв'язання задач лінійного програмування та змішаного цілочислового програмування. Зауважимо, що за кожний послідовний запуск функції GLPK алгоритм або знаходить ще один максимальний k-плекс, або закінчує свою роботу, якщо всі n_{sol} k-плексів уже знайдено. У випадку, якщо кількість розв'язків у задачі (3.1.11) – (3.1.13) є більшою за ncalls, то в результаті роботи алгоритму отримаємо ncalls максимальних k-плексів. Для того, щоб продовжити пошук нових k-плексів, потрібно збільшити величину ncalls.

Алгоритм перевірено на ряді тестових прикладів. Дані для графа читаються з текстового файла у DIMACS-форматі. Нижче наведемо результати обчислювальних експериментів для пошуку всіх максимальних k-плексів для графа «1zc.128» (128 вершин, 2240 ребер) при різних значеннях $k = \overline{1,8}$ (табл. 3.1.1) та для пошуку всіх максимальних клік у чотирьох графах типу «1zc» з 128, 256, 512, 1024 вершинами (табл. 3.1.2). Кількості вершин та ребер для графів «1zc.256», «1zc.512» та «1zc.1024» наведені в першій колонці табл. 3.1.2. Максимальний за розмірами граф «1zc.1024» містить 1024 вершин та 33280 ребер. Обчислювальні експерименти здійснювались на комп'ютері з процесором Intel Core i5-9400f: частота 2.9 Ггц та оперативна пам'ять 16 Гб.

В табл. 3.1.2 наведено кількості знайдених розв'язків задачі (3.1.11) – (3.1.13) (колонка n_{col}); найменші (колонка $t_{\min}$) та найбільші (колонка $t_{\max}$) значення часу, який було затрачено на розв'язання однієї із задач (3.1.11) – (3.1.13), (3.1.11) – (3.1.13), (3.1.35); середній час (колонка t_{avr}) визначався як сумарний час,

затрачений програмою GLPK на розв'язання задач (3.1.11) – (3.1.13), (3.1.11) – (3.1.13), (3.1.35), розділений на величину ($n_{sol}+1$).

Табл. 3.1.2. Затрати часу (в секундах) на пошук n_{sol} максимальних k -плексів у графі «1zc.128»: $k=\overline{1,8}$, ncalls $=1001$

k	n_{sol}	$t_{\min}$	$t_{\max}$	t_{avr}		k	n_{sol}	$t_{\min}$	$t_{\max}$	t_{avr}
1	2	1.09	1.11	1.10		5	630	5.93	49.04	14.25
2	2	2.61	2.71	2.66		6	56	4.68	17.80	7.85
3	2	10.32	11.39	10.84		7	196	2.74	9.73	4.66
4	728	2.92	16.08	7.14		8	>1000	19.99	218.16	58.61

Табл. 3.1.3. Затрати часу (в секундах) на пошук максимальних клік у графах типу «1zc» з 128, 256, 512, 1024 вершинами: ncalls $=3$

$n,\lvert E\rvert$	Знайдені максимальні кліки	t_{avr}
128, 2240	$\{64,96,112,120,124,126,127,128\}$ $\{1,2,3,5,9,17,33,65\}$	1.10
256, 5632	$\{1,2,3,5,9,17,33,65,129\}$ $\{128,192,224,240,248,252,254,255,256\}$	13.43
512, 13824	{1, 2, 3, 5, 9, 17, 33, 65, 129, 257} {256, 384, 448, 480, 496, 504, 508, 510, 511, 512}	234.92
1024, 33280	{1, 2, 3, 5, 9, 17, 33, 65, 129, 257, 513} {512, 768, 896, 960, 992, 1008, 1016, 1020, 1022, 1023, 1024}	3808.30

В табл. 3.1.3 наведено по дві знайдені максимальні кліки для чотирьох графів типу «1zc» та доведено, що інших максимальних клік немає, так як кількість знайдених клік менша за ncalls $=3$.

3.1.8 Висновки

У підрозділі побудовано квадратичне формулювання оптимізаційної задачі (3.1.7) – (3.1.9) для знаходження максимального k-плекса у неорієнтованому графі. Показано, що її можна отримати із відомої лінійної моделі у формі задачі лінійного булевого програмування (3.1.11) – (3.1.13), домножуючи обмеження на невід'ємні змінні для кожної із вершин графа. Спорідненість понять k-плекса та кліки дає підстави рекомендувати для її розв'язання запропонований Н. З. Шором підхід, пов'язаний з лагранжевими оцінками та використанням надлишкових обмежень. Цей підхід дав ряд важливих теоретичних результатів для задачі про максимальну незалежну множину вершин графа [9, 16].

Із прикладів квадратичних задач (3.1.22) – (3.1.24) та (3.1.26) – (3.1.28) для знаходження максимальних 2-плексів графів G_1 та G_2 видно, що функціонально надлишкові обмеження виду (3.1.14) та (3.1.15) відіграють значну роль для поліпшення точності лагранжевих двоїстих оцінок. Так, з їх допомогою лагранжеву двоїсту оцінку ψ^* можна зробити точною для $\rho_2(G_1)$ та $\rho_2(G_2)$. Однак, це не означає, що для будь-якої квадратичної задачі (3.1.11) – (3.1.13) за допомогою обмежень виду (3.1.14) та (3.1.15) можна домогтися того, щоб лагранжева двоїста оцінка була точною. Для підвищення точності лагранжевих двоїстих оцінок можна використовувати й інші способи побудови функціонально надлишкових обмежень [9, 16].

Розроблено алгоритм пошуку всіх максимальних k-плексів для неорієнтованого графа. В основі алгоритму лежить послідовне додавання до задачі лінійного булевого програмування додаткового обмеження, яке відсікає вже знайдені максимальні k-плекси. Алгоритм був реалізований на мові Octave за допомогою GLPK (GNU Linear Programming Kit). Наведено результати обчислень для пошуку всіх максимальних k-плексів при $k = \overline{1,8}$ для графа «1zc.128» та всіх максимальних клік для графів типу «lzc» з 128, 256, 512, 1024 вершинами.

Робота виконана за підтримки фундаментальної теми Національної академії наук України (проєкт № 0117U000327 «Розробити субградієнтні алгоритми розв'язання багатоекстремальних квадратичних оптимізаційних задач», 2017–2021 рр.), а також гранту CRDF Global та МОН України (G-202102-68020 «Методи оптимізації зі зменшенням ризиків для розміщення об'єктів у виробництві відновлюваної енергії», 2021-2022), за яким опубліковано статтю [17].

Список літератури

1. Seidman S.B., Foster B.L. A graph theoretic generalization of the clique concept. J. of Math. Sociology. 1978. Vol 6. Pp. 139–154.
2. Balansundaram B., Butenko S., Hicks I.V. Clique Relaxations in Social Network Analysis: The Maximum k-plex Problem. Operations Research. 2011. Vol 59. No 1. Pp. 133–142.
3. Guo J., Komusiewicz C., Niedermeier R., Uhlmann J. A more relaxed model for graph-based data clustering: s-plex cluster editing. SIAM J. Discrete Math. 2010. Vol 24.No 4. P. 1662–1683.
4. McClosky B., Hicks I.V. Combinatorial algorithms for the maximum k-plex problem. J. of Combinatorial Optimization. 2012. Vol 23. No 1. Pp. 29–49.
5. Pattillo J., Veremyev A., Butenko S., Boginski V. On the maximum quasi-clique problem. Discrete Applied Mathematics. 2013. Vol 161. No 1–2. Pp. 244–257.
6. Стецюк П.І., Бардадим Т.О., Ляшко В.І. Квадратична задача для максимального k-плекса в неорієнтованому графі. Журнал обчислювальної та прикладної математики. 2017. №1(124). С. 80–87.
7. Стецюк П.І., Ляшко В.І., Бардадим Т.О. Властивості квадратичної задачі про максимальний k-плекс у неорієнтованому графі. Наукові записки НаУКМА. 2017. Том 198. Комп'ютерні науки. 2017. №1(124). С. 80–87.
8. Стецюк П.І., Хом'як О.М. Застосування пакету GLPK для знаходження всіх розв'язків задачі про k-плекс. Матеріали XXIII Міжнародного науково-практичного семінару імені А.Я.

Петренюка "Комбінаторні конфігурації та їхні застосування", присвяченого 70-річчю Льотної академії Національного авіаційного університету (Запоріжжя – Кропивницький, 13-15 травня 2021 р.) / за ред. Г.П. Донця. Кропивницький: ПП «Ексклюзив-Систем», 2021. С. 174–179.

9. Shor N. Z. Nondifferentiable Optimization and Polynomial Problems. London/Boston/Dordrecht:Kluwer Academic Publishers, 1998. 394 p.
10. Shor N.Z., Stetsyuk P.I. Dual Solution of Quadratic-Type Problems by r-algorithm (subroutine DSQTPr). Abstracts of the Second International Workshop "Recent Advances in Non-Differentiable Optimization" (October 1-4, 2001, Kyiv, Ukraine). P. 36.
11. Eaton J.W., Bateman D., Hauberg S. GNU Octave Manual Version 3. Network Theory Ltd, 2008. 568 p.
12. Стецюк П.І., Журбенко М.Г., Сергієнко І.В. та інші. Нові мережево-орієнтовані методики для інформаційного аналізу великих масивів даних. Звіт про науково-дослідну работу М/163-2006, № держ. реєстрації 0106U010005. Київ: Ін-т кібернетики ім. В.М. Глушкова НАН України, 2006. 112 с.
13. Стецюк П.И. О функционально избыточных ограничениях для булевых оптимизационных задач квадратичного типа. Кибернетика и системный анализ. 2005. № 6. С. 168–172.
14. Стецюк П.И. Новые модели квадратичного типа для задачи о максимальном взвешенном разрезе графа. Кибернетика и системный анализ. 2006. № 1. С. 63–75.
15. Стецюк П.І., Слабоспицька О.О., Ушакова О.О. Максимальні незалежні множини вершин графа та їх застосування в керуванні проектами. Питання прикладної математики і математичного моделювання. Д.: РВВ ДНУ. 2016.
16. Стецюк П.И. Двойственные оценки в квадратичных экстремальных задачах. Кишинэу: Эврика. 2018. 504 с.
17. Стецюк П.І., Хом'як О.М., Блохін Є.А., Супрун А.А. Оптимізаційні задачі для максимального k-плекса. Кібернетика та системний аналіз. 2022. Т. 58. № 4. С. 46–58.

3.2 ПОШУК ДЕФЕКТІВ У РЕГУЛЯРНИХ 3D-СТРУКТУРАХ

П. І. Стецюк, В. В. Савицький, В. О. Жидков

Анотація. Розглядаються оптимізаційні задачі для знаходження найкращих по L_p-нормі параметрів регулярних 3D-структур і методи найменших модулів та найменших квадратів для їх розв'язання. Показано, що при відновленні параметрів 3D-структур з дефектами метод найменших модулів стійкіший, ніж метод найменших квадратів. Наведено результати обчислювальних експериментів для програмних реалізацій методів на основі *r*-алгоритму Шора.

Abstract. Optimization problems for finding best L_p-norm parameters of regular 3D-structures and least module and least square methods for their solving are considered. It is shown that the least module method is more stable than the least squares method, when restoring the parameters of 3D-structures with defects. The results of computational experiments for software implementations of methods based on Shor's *r*-algorithm are presented.

3.2.1 Вступ

Регулярні зображення з дефектами (рис. 3.2.1) характерні при неруйнуючому контролі якості (НКЯ) тонкостінних багатошарових композиційних матеріалів за допомогою методів лазерної інтерферометрії, таких, як метод голографічної інтерферометрії, метод спекл-інтерферометрії та метод ширографії [1].

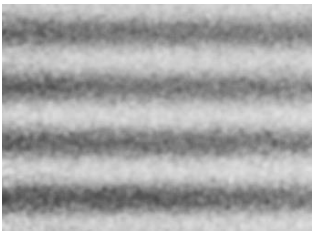
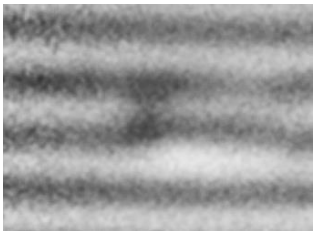
а)  б) 

Рис. 3.2.1. Приклади зображень: a) регулярне; b) регулярне з дефектною областю – порушення регулярності в центрі

Методи лазерної інтерферометрії для НКЯ передбачають вимірювання переміщень або деформацій точок поверхні досліджуваних об'єктів, які виникають у результаті термічного або механічного навантаження. При НКЯ елементів конструкцій, які мають періодичну (регулярну) 3D-структуру, поле вимірюваних методами лазерної інтерферометрії величин (переміщення, деформації) повинне також мати періодичну структуру.

Якщо при виготовленні або експлуатації таких елементів конструкцій виникають дефекти (наприклад, відсутність з'єднання, наявність тріщин, пор, вм'ятин тощо), то в таких місцях порушується регулярність поля вимірюваних величин, і область дефекту визначається оператором (спостерігачем) як місце, у якому є очевидним це порушення.

Потрібно автоматизувати процес визначення місця розташування дефектів у відповідальних елементах конструкцій, щоб знизити вплив людського фактора при неруйнуючому контролі якості за допомогою зазначених вище методів.

У розділі розглянемо оптимізаційні задачі для регулярних 3D-структур, які введені авторами у роботі [2] і мають місце при аналізі регулярних зображень із дефектами, і методи їх розв'язання на основі методу Ньютона та алгоритмів негладкої оптимізації.

Матеріал розділу викладений у такому порядку. У параграфі 3.2.2 визначено регулярні 3D-структури та пов'язані з нею дефекти. У параграфі 3.2.3 сформульовано оптимізаційні задачі для знаходження

найкращих за L_p-нормою параметрів регулярних 3D-структур і досліджено їхні властивості. У параграфі 3.2.4 описано методи найменших квадратів і найменших модулів для знаходження параметрів регулярних структур.

Показано, що метод найменших квадратів не здатний відновити їхні параметри при наявності хоча б одного дефекту, а метод найменших модулів є стійким до знаходження параметрів регулярних 3D-структур з однією та декількома областями з дефектами. У параграфі 3.2.5 описано дві загальні нерівності для суми квадратів двох наборів чисел, перша з яких неявно використовувалася у параграфі 3.2.2 при визначенні базисної регулярної 3D-структури.

3.2.2 Регулярні 3D-структури, їхні параметри та дефекти

3D-структурою будемо називати трійку $\{A,u,v\}$, де A – $m\times n$-матриця, така що $A=\{a_{ij}\}_{i=1,\dots,m}^{j=1,\dots,n}\in R^{m\times n}$, $u\in R^m$ і $v\in R^n$. Параметрами 3D-структури $\{A,u,v\}$ будемо називати m-мірний вектор u і n-вимірний вектор v, їхні компоненти будуть визначати значення елементів матриці $A\in R^{m\times n}$.

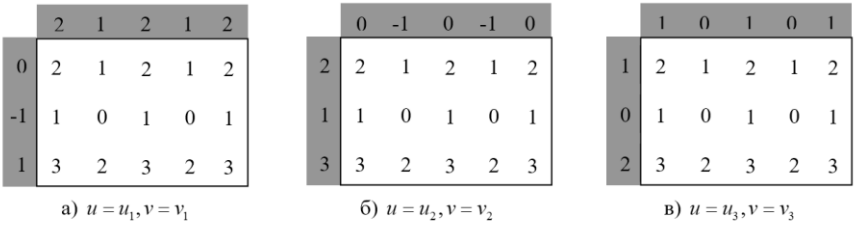
Означення 3.2.1. 3D-структура $\{A,u,v\}$ називається регулярною, якщо матриця $A\in R^{m\times n}$, а вектори $u\in R^m$ і $v\in R^n$ є такими, що $a_{ij}=u_i+v_j$, $i=1,\dots,m$, $j=1,\dots,n$.

Одна із властивостей регулярної 3D-структури визначається наступним твердженням.

Лема 3.2.1. Якщо $\{A,u,v\}$ – регулярна 3D-структура, то регулярною буде й 3D-структура $\{A,\tilde{u},\tilde{v}\}$, де $\tilde{u}_i=u_i+t$, $i=1,\dots,m$, $\tilde{v}_j=v_j-t$, $j=1,\dots,n$, $\forall t\in R$, $t\neq 0$.

Лема 3.2.1 означає, що для регулярної 3D-структури параметри u та v визначені неоднозначно. Так, наприклад, на рис. 3.2.2 наведено три різні набори параметрів для однієї й тієї ж 3×5-матриці

$A = \begin{vmatrix} 2 & 1 & 2 & 1 & 2 \\ 1 & 0 & 1 & 0 & 1 \\ 3 & 2 & 3 & 2 & 3 \end{vmatrix}$, яка визначає регулярну 3D-структуру. Першій регулярній 3D-структурі $\{A,u,v\}$ відповідають параметри $u = u_1, v = v_1$, другій – $u = u_2, v = v_2$, третій – $u = u_3, v = v_3$.



а) $u = u_1, v = v_1$ б) $u = u_2, v = v_2$ в) $u = u_3, v = v_3$

Рис. 3.2.2. Неоднозначні регулярні 3D-структури:
а) $u_1 = (0,-1,1)^T$, $v_1 = (2,1,2,1,2)^T$; б) $u_2 = (2,1,3)^T$, $v_2 = (0,-1,0,-1,0)^T$;
в) $u_3 = (1,0,2)^T$, $v_3 = (1,0,1,0,1)^T$

Щоб параметри регулярної 3D-структури були визначені однозначно, досить зафіксувати величину t з тих або інших міркувань. Так, наприклад, її значення можна вибрати таким, щоб вираз $\|u + te_m\|^2 + \|v - te_n\|^2$ був мінімальним. Тут e_m і e_n – m-мірний і n-мірний вектори, усі компоненти яких дорівнюють одиниці. Це призводить до мінімізації одномірної функції

$$f(t) = \|u + te_m\|^2 + \|v - te_n\|^2 =$$

$$= \sum_{i=1}^{m} u_i^2 + \sum_{j=1}^{n} v_j^2 + 2\left(\sum_{i=1}^{m} u_i - \sum_{j=1}^{n} v_j\right)t + (m+n)t^2. \qquad (3.2.1)$$

Вона досягає мінімального значення f^* при $t = t^*$, які визначаються за формулами

$$f^* = f(t^*) = \sum_{i=1}^{m} u_i^2 + \sum_{j=1}^{n} v_j^2 - \frac{\left(\sum_{i=1}^{m} u_i - \sum_{j=1}^{n} v_j\right)^2}{m+n}, \quad t^* = \frac{\sum_{j=1}^{n} v_j - \sum_{i=1}^{m} u_i}{m+n}. \quad (3.2.2)$$

За допомогою формул (3.2.1) і (3.2.2) можна однозначно визначити параметри регулярної 3D-структури. З виразу для f^* у (3.2.2) випливає, що зменшити суму квадратів компонент векторів u і v не можна, якщо для них виконана умова $\sum_{i=1}^{m} u_i = \sum_{j=1}^{n} v_j$. Саме цю умову поставимо в основу однозначної (по параметрах) регулярної 3D-структури.

Означення 3.2.2. Регулярну 3D-структуру $\left\{A, u^*, v^*\right\}$ будемо називати базисною, якщо її параметри $u^* \in R^m$ та $v^* \in R^n$ – такі, що $\sum_{j=1}^{m} u_j^* = \sum_{i=1}^{n} v_i^*$.

Справедливе таке твердження.

Лема 3.2.2. Якщо $\{A, u, v\}$ – регулярна 3D-структура, то регулярна 3D-структура $\{A, u^*, v^*\}$ буде базисною, якщо

$$u^* = u + t^* e_m; \quad v^* = v - t^* e_n, \quad \text{де} \quad t^* = \frac{\sum_{j=1}^{n} v_j - \sum_{i=1}^{m} u_i}{m+n}. \quad (3.2.3)$$

Якщо $\{A, u^*, v^*\}$ – базисна регулярна 3D-структура, то 3D-структура $\{A, u, v\}$ буде регулярною, якщо

$$u = u^* + te_m, \quad v = v^* - te_n \quad (3.2.4)$$

для довільного $t \in R$.

Лема 3.2.2 дає нам формули (3.2.3) і (3.2.4), які зв'язують між собою параметри базисної й звичайної регулярних 3D-структур. Так, наприклад, базисною регулярною 3D-структурою для 3×5-матриці

$A=\begin{vmatrix}2&1&2&1&2\\1&0&1&0&1\\3&2&3&2&3\end{vmatrix}$ є третя регулярна 3D-структура з рис. 3.2.2, параметри якої $u=u_3=(1,0,2)^T$ та $v=v_3=(1,0,1,0,1)^T$. Для неї $\sum_{i=1}^{3}u_i=\sum_{j=1}^{5}v_j=3$, отже $u^*=(1,0,2)^T$ і $v^*=(1,0,1,0,1)^T$. Щоб від першої та другої регулярних 3D-структур з рис. 3.2.2 перейти до третьої, досить у формулах (3.2.3) використовувати $u=u_1$, $v=v_1$, $t^*=t_1^*=-1$ або $u=u_2$, $v=v_2$, $t^*=t_2^*=1$. За допомогою формули (3.2.4) і параметрів базисної регулярної 3D-структури для 3×5-матриці A можна побудувати не тільки першу, а й другу регулярні 3D-структури (їм у формулі (3.2.4) відповідають значення $t=t_1=1$ та $t=t_2=-1$), а також низку інших 3D-структур, що задовольняють ті або інші властивості.

Наприклад, при $t=t_4=0.5$ одержуємо регулярну 3D-структуру з параметрами $u=u_4=(0.5,-0.5,1.5)^T$ і $v=v_4=(1.5,0,1.5,0,1.5)^T$, тобто таку, що останні компоненти векторів u і v однакові.

Означення 3.2.3. Елементарним дефектом у регулярній 3D-структурі $\{A,u,v\}$ будемо називати таку пару індексів i,j, $i\in1,\ldots,m$, $j\in1,\ldots,n$, для яких $a_{ij}\neq u_i+v_j$.

Для регулярних 3D-структур з дефектами будемо дотримуватися таких назв. Якщо регулярна 3D-структура має один елементарний дефект, то її будемо називати регулярною з одним дефектом. Якщо регулярна 3D-структура має k елементарних дефектів ($k>1$), то її будемо називати регулярною з k дефектами. Приклади регулярних 3D-структур з одним і трьома дефектами 3×5-матриці $\tilde{A}$ наведено на рис. 3.2.3, де в «рамки» узяті ті елементи матриці $\tilde{A}$ з індексами i,j, де має місце елементарний дефект $\tilde{a}_{ij}\neq u_i+v_j$. Це зроблене на прикладі тієї ж 3×5-матриці A, що й на рис. 3.2.1.

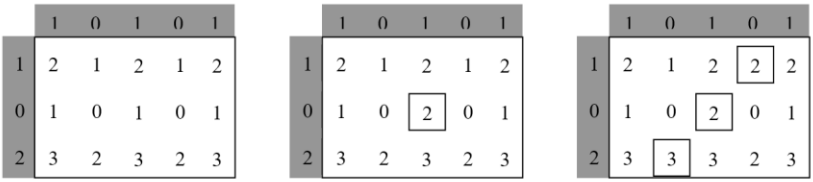


Рис. 3.2.3. Регулярні 3D-структури:
а) базисна; б) з одним дефектом $\tilde{a}_{23} = a_{23} + 1$;
в) з трьома дефектами $\tilde{a}_{14} = a_{14} + 1$, $\tilde{a}_{23} = a_{23} + 1$, $\tilde{a}_{32} = a_{32} + 1$.

Нам буде досить визначення 3.2.3, але слід зазначити, що для реальних задач поняття «дефекту» може потребувати більш складного визначення. У них для регулярних зображень під «дефектними» розуміються деякі множини пар індексів i, j, у яких порушена умова $a_{ij} \neq u_i + v_j$.

Як правило, ці множини задають зв'язні області, але в принципі вони можуть задавати й незв'язні області.

3.2.3 Задачі для пошуку найкращих параметрів

Нижче розглянемо формулювання оптимізаційних задач для знаходження найкращих параметрів регулярної 3D-структури й базисної регулярної 3D-структури, де під «найкращими параметрами» будемо розуміти такі значення векторів x^* і y^*, що коефіцієнти $a^*_{ij} = x^*_i + y^*_j$ мінімально (зокрема, в евклідовій і манхетенівській нормах) відхиляються від коефіцієнтів деякої заданої матриці A.

Задача 1. Є $m \times n$-матриця A. Для регулярної 3D-структури $\{A, x^*, y^*\}$ потрібно знайти такі вектори $x^* \in R^m$ та $y^* \in R^n$, щоб

коефіцієнти $m \times n$-матриці A^* мінімально відхилялися від коефіцієнтів $m \times n$-матриці A.

Пошук найкращих параметрів для регулярної 3D-структури можна забезпечити за допомогою безумовної задачі мінімізації негладкої опуклої функції: знайти

$$f_p^* = f_p(x_p^*, y_p^*) =$$

$$= \min_{x \in R^m,\ y \in R^n} \left\{ f_p(x,y) = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} \left| a_{ij} - x_i - y_j \right|^p \right)^{1/p} \right\}, \qquad (3.2.5)$$

де $|\cdot|$ – модуль числа, а скалярна величина p така, що $1 \le p \le 2$. Розв'язок задачі (3.2.5) дає регулярну 3D-структуру $\{A^*, x_p^*, y_p^*\}$, а коефіцієнти матриць A і A^* будуть мінімально різнитися в\по так званій L_p-нормі, тобто, коли норма вектора $z = (z_1, \ldots, z_N)^T$ визначена в такий спосіб: $\|z\|_p = \left(\sum_{i=1}^{N} |z_i|^p \right)^{1/p}$, де $p \ge 1$. Якщо $p = 2$, то коефіцієнти матриць A і A^* будуть мінімально відхилятися в евклідовій нормі, якщо $p = 1$ – то в манхетенівській нормі.

Крім того, можна розглядати й інші значення величини $p: 1 < p < 2$. Її вибором визначається той або інший метод для відновлення параметрів регулярних 3D-структур, наприклад, метод найменших квадратів ($p = 2$) або метод найменших модулів ($p = 1$).

Задача (3.2.5) має багато розв'язків, оскільки значення параметрів регулярної 3D-структури $\{A^*, x_p^*, y_p^*\}$ визначені неоднозначно (див. лему 3.2.1). Однозначно вони будуть визначені тоді, коли регулярна 3D-структура $\{A^*, x_p^*, y_p^*\}$ буде ще й базисною.

Параметри базисної регулярної 3D-структури можна визначити в такий спосіб: спочатку розв'язати задачу (3.2.5), а потім для знаходження u^* та v^* використати формули (3.2.3), де $u = x_p^*$ і

$v = y_p^*$. Але це ж саме можна зробити за допомогою свого формулювання оптимізаційної задачі, яка буде розрахована на пошук найкращих параметрів базисної регулярної 3D-структури.

Задача 2. Є $m \times n$-матриця A. Для базисної регулярної 3D-структури $\{A^{**}, x^{**}, y^{**}\}$ потрібно знайти такі вектори $x^{**} \in R^m$ та $y^{**} \in R^n$, щоб коефіцієнти $m \times n$-матриці A^{**} мінімально відхилялися від коефіцієнтів $m \times n$-матриці A.

Пошук найкращих параметрів для базисної регулярної 3D-структури $\{A^{**}, x^{**}, y^{**}\}$ можна забезпечити, якщо задачу (3.2.5) доповнити єдиним обмеженням

$$\sum_{i=1}^{m} x_i - \sum_{j=1}^{n} y_j = 0, \qquad (3.2.6)$$

яке задає умови на параметри базисної 3D-структури з леми 3.2.2. Задача (3.2.5) – (3.2.6) є задачею опуклого нелінійного програмування. Їй буде відповідати таке ж, як і для задачі (3.2.5), оптимальне значення f_p^*, а її оптимальний розв'язок x_p^{**} і y_p^{**} буде визначати найкращі по L_p-нормі параметри базисної регулярної структури $\{A^{**}, x^{**}, y^{**}\}$. Якщо $p = 2$, то для знаходження параметрів базисної регулярної 3D-структури одержимо метод найменших квадратів, а якщо $p = 1$ – метод найменших модулів.

Оптимізаційні задачі (3.2.5) і (3.2.5) – (3.2.6) можна розв'язувати за допомогою стандартного програмного забезпечення для розв'язання задач нелінійного програмування. Однак тут є деякі особливості, які пов'язані з представленням задач (3.2.5) і (3.2.5) – (3.2.6) у формі задач опуклого програмування.

Так, наприклад, задачу (3.2.5) можна звести до такої задачі опуклого програмування: знайти

$$\left(f_p^*\right)^p = \left(f_p(z_p^*, x_p^*, y_p^*)\right)^p =$$

$$= \min_{z \in R^{m \times n}, x \in R^m, y \in R^n} \left\{ (f_p(z,x,y))^p = \sum_{i=1}^{m} \sum_{j=1}^{n} z_{ij}^p \right\}, \qquad (3.2.7)$$

при обмеженнях

$$-z_{ij} - x_i - y_j \le -a_{ij}, \quad i = 1,\ldots,m,\ j = 1,\ldots,n, \qquad (3.2.8)$$

$$-z_{ij} + x_i + y_j \le a_{ij}, \quad i = 1,\ldots,m,\ j = 1,\ldots,n, \qquad (3.2.9)$$

яка при $p = 1$ перетворюється у задачу лінійного програмування, а при $p = 2$ – у задачу квадратичного програмування. Щоб задачу (3.2.5) – (3.2.6) звести до задачі опуклого програмування, досить до задачі (3.2.7) – (3.2.9) додати ще лінійне обмеження (3.2.6). Особливістю задач (3.2.7) – (3.2.9) і (3.2.7) – (3.2.9), (3.2.6) є те, що цільова функція (3.2.7) визначена тільки для невід'ємних значень змінних $z_{ij} \ge 0$, які задовольняють обмеження (3.2.8), (3.2.9).

Дійсно, якщо для деякої пари індексів i та j скласти нерівності (3.2.8) і (3.2.9), то одержимо нерівність $-2z_{ij} \le 0$, яка рівносильна $z_{ij} \ge 0$. Тому при таких значеннях p, коли $1 < p < 2$, для розв'язання задач (3.2.7) – (3.2.9) і (3.2.7) – (3.2.9), (3.2.6) застосовні тільки ті ітераційні методи, які працюють із припустимими точками для системи обмежень (3.2.8) – (3.2.9). Крім того, навіть при порівняно невеликих значеннях m і n задача (3.2.7) – (3.2.9) буде мати велику кількість змінних $N = m \times n + m + n$ і обмежень $M = 2 \times m \times n$.

Перспективнішим є розв'язання задачі (3.2.5) за допомогою субградієнтних методів для мінімізації негладких опуклих функцій [3, 4]. При цьому кількість змінних $N = m + n$, а субградієнт опуклої функції $f_p(x,y)$ обчислюється за формулою

$$g_{f_p}(x,y) =$$

$$= \left(f_p(x,y) \right)^{1-p} \begin{cases} -\sum_{j=1}^{n} \operatorname{sgn}\left(a_{ij} - x_i - y_j \right) \left| a_{ij} - x_i - y_j \right|^{p-1}, i = \overline{1,m}, \\ -\sum_{i=1}^{m} \operatorname{sgn}\left(a_{ij} - x_i - y_j \right) \left| a_{ij} - x_i - y_j \right|^{p-1}, j = \overline{1,n}, \end{cases} \quad (3.2.10)$$

яку легко реалізувати для матлабо-подібних мов програмування. Цей шлях також підходить і для задачі (3.2.5) – (3.2.6), бо вона зводиться до такої задачі мінімізації негладкої опуклої функції: знайти

$$F_p^* = F_p\left(x_p^{**}, y_p^{**} \right) = \min_{x \in R^m,\ y \in R^n} \left\{ F_p\left(x, y \right) = \right.$$

$$\left. = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} \left| a_{ij} - x_i - y_j \right|^p \right)^{\frac{1}{p}} + M \left| \sum_{i=1}^{m} x_i - \sum_{j=1}^{n} y_j \right|^q \right\}, \quad (3.2.11)$$

де скалярні величини p і q такі, що $1 \le p, q \le 2$, а скалярна величина $M > 0$. Слід зазначити, що для розв'язання задачі (3.2.5) – (3.2.6) можна використовувати також субградієнтні методи із проекцією на лінійну рівність (3.2.6). Так, наприклад, при застосуванні *r*-алгоритмів це призводить до певного способу установки початкової матриці B_0 і до наступної її корекції таким чином, щоб для кожної точки ітераційного процесу виконувалася рівність (3.2.6).

Отже, за допомогою розв'язання задачі (3.2.5) можна знаходити параметри регулярних 3D-структур, а за допомогою розв'язання задачі (3.2.5) – (3.2.6) можна відновлювати параметри базисних регулярних 3D-структур.

Справедливе таке твердження.

Теорема 3.2.1. Якщо 3D-структура $\{A, u, v\}$ – регулярна, то в задачі (3.2.5) розв'язок x_p^* і y_p^* такий, що $f_p^* = f(x_p^*, y_p^*) = 0$, а 3D-структура $\{A, x_p^*, y_p^*\}$ є регулярною. Якщо регулярна 3D-структура $\{A, u^*, v^*\}$ є базисною, то в задачі (3.2.5) – (3.2.6) розв'язок x_p^{**} і y_p^{**} такий, що $f_p^* = f(x_p^{**}, y_p^{**}) = 0$, а $u^* = x_p^{**}$ і $v^* = y_p^{**}$.

Якщо до регулярної 3D-структури додати один або кілька елементарних дефектів, то $f_p^* \neq 0$ як для задачі (3.2.5), так і для задачі (3.2.5) – (3.2.6). При цьому вибір параметра p суттєво впливає та те, зможемо чи не зможемо ми знайти параметри регулярних 3D-структур. Це ілюструють обчислювальні експерименти по відновленню параметрів $u^* = (1,0,2)^T$ і $v^* = (1,0,1,0,1)^T$ для базисної регулярної 3D-структури $\left\{A^*, u^*, v^*\right\}$ за 3×5-матрицями $A_1 = \begin{vmatrix} 2 & 1 & 2 & 1 & 2 \\ 1 & 0 & 1+1 & 0 & 1 \\ 3 & 2 & 3 & 2 & 3 \end{vmatrix}$ і $A_2 = \begin{vmatrix} 2 & 1 & 2 & 1+1 & 2 \\ 1 & 0 & 1+1 & 0 & 1 \\ 3 & 2+1 & 3 & 2 & 3 \end{vmatrix}$, які містять один і три елементарні дефекти відповідно (див. рис. 3.2.3). Їхні результати для десяти значень $p \in [1,2]$ відображені в табл. 3.2.1, де наведено значення p (перший стовпчик), оптимальні значення цільових функцій (стовпці $f_p\left(x_p^{**}, y_p^{**}\right)$) і відхилення знайдених параметрів x_p^{**} і y_p^{**} від точних $u^* = (1,0,2)^T$ і $v^* = (1,0,1,0,1)^T$ (стовпчики $\left\|u^* - x_p^{**}\right\|$ і $\left\|v^* - y_p^{**}\right\|$). Тут за допомогою r-алгоритму (програма ralgb5 [5]) розв'язувалася оптимізаційна задача (3.2.11) при $M = 1$ і $q = 1$.

Табл. 3.2.1. Відновлення параметрів 3D-структур з дефектами при різних p

p	Один дефект ($A = A_1$)			Три дефекти ($A = A_2$)		
	$f_p(x_p^{**}, y_p^{**})$	$\left\|u^* - x_p^{**}\right\|$	$\left\|v^* - y_p^{**}\right\|$	$f_p(x_p^{**}, y_p^{**})$	$\left\|v^* - y_p^{**}\right\|$	$\left\|v^* - y_p^{**}\right\|$
1.00	1.00000	0.00000	0.00000	3.00000	0.00000	0.00000
1.05	1.00000	0.00000	0.00000	2.84709	0.00000	0.00000
1.06	1.00000	0.00000	0.00001	2.81912	0.00001	0.00001
1.08	0.99999	0.00004	0.00016	2.76550	0.00011	0.00021

1.10	0.99991	0.00021	0.00089	2.71461	0.00063	0.00118
1.20	0.99472	0.00693	0.02762	2.48527	0.01968	0.03653
1.40	0.94620	0.04579	0.13764	2.09220	0.09757	0.18108
1.60	0.87096	0.09568	0.21940	1.79311	0.15558	0.28875
1.80	0.79586	0.14130	0.27027	1.57515	0.19226	0.35681
2.00	0.73030	0.17854	0.30334	1.41421	0.21651	0.40182

З табл. 3.2.1 видно, що при $p=1$ (відповідає методу найменших модулів) параметри $u^*=(1,0,2)^T$ та $v^*=(1,0,1,0,1)^T$ відновлюються точно і при одному, і при трьох елементарних дефектах. Про це свідчать оптимальні значення функцій, які для одного та трьох дефектів повинні дорівнювати $\|A_1-A^*\|=1$ та $\|A_2-A^*\|=3$ відповідно.

Схожа поведінка спостерігається також для значень p, які близькі до одиниці. Але зовсім інша картина має місце при $p=2$ (відповідає методу найменших квадратів) і значеннях p, які близькі до двійки. Тут говорити про досить точне відновлення параметрів не доводиться, тому що відхилення знайдених параметрів x_2^{**} і y_2^{**} від точних значень параметрів u^* і v^* вже досить великі як при наявності одного, так і при наявності трьох елементарних дефектів.

3.2.4 Методи найменших квадратів та найменших модулів

Метод найменших квадратів (МНК) і метод найменших модулів (МНМ) є важливими окремими випадками методів розв'язування оптимізаційних задач (3.2.5) і (3.2.5) – (3.2.6). Так, МНК отримуємо, якщо вибрати $p=2$, а МНМ – якщо вибрати $p=1$. Якщо методи пов'язані з розв'язуванням задачі (3.2.5), то вони дозволяють знаходити найкращі параметри регулярних 3D-структур, а якщо з розв'язуванням задачі (3.2.5) – (3.2.6), то – найкращі параметри базисних регулярних 3D-структур. Для обох методів виконується

теорема 3.2.1, а значить МНМ і МНК дозволяють відновлювати параметри базисних регулярних 3D-структур.

Це можна зробити двома способами: перший – знайти один з розв'язків задачі (3.2.5) і застосувати формулу (3.2.3) з леми 3.2.2, другий – знайти єдиний розв'язок задачі (3.2.5) – (3.2.6).

Для регулярних 3D-структур з одним або декількома дефектами поведінка МНК і МНМ суттєво відрізняється. Так, якщо в МНК немає шансів відновити параметри регулярних 3D-структур з дефектами, то в МНМ такі шанси дуже великі, що підтверджують результати обчислювальних експериментів з табл. 3.2.1.

Відповідні до методів МНК і МНМ оптимізаційні задачі можна спростити. Так, наприклад, для МНМ задача опуклого програмування (3.2.7) – (3.2.9) переходить у таку задачу лінійного програмування:

$$f_1^* = f_1\left(z^*, x^*, y^*\right) = \min_{z \in R^{m \times n}, x \in R^m, y \in R^n} \left\{ f_1(z, x, y) = \sum_{i=1}^{m} \sum_{j=1}^{n} z_{ij} \right\} \quad (3.2.12)$$

за обмежень (3.2.8) – (3.2.9) (для регулярних структур), або (3.2.8) – (3.2.9), (3.2.6) для (базисних регулярних структур).

Для МНМ задача опуклого програмування (3.2.7) – (3.2.9) переходить у задачу квадратичного програмування

$$\left(f_2^*\right)^2 = \left(f_2\left(z^*, x^*, y^*\right)\right)^2 =$$

$$= \min_{z \in R^{m \times n},\ x \in R^m,\ y \in R^n} \left\{ \left(f_2\left(z^*, x^*, y^*\right)\right)^2 = \sum_{i=1}^{m} \sum_{j=1}^{n} z_{ij}^2 \right\}, \quad (3.2.13)$$

при обмеженнях (3.2.8) – (3.2.9) (для регулярних структур), або (3.2.8) – (3.2.9), (3.2.6) для (базисних регулярних структур).

Для задач лінійного і квадратичного програмування немає ніяких незручностей, пов'язаних з областю визначення цільової функції, які обговорювалися для задачі (3.2.7) – (3.2.9). Крім того, їх легко доповнювати лінійними обмеженнями, які можуть задавати додаткові умови на ті параметри регулярних 3D-структур, які потрібно знайти.

Незважаючи на те, що зазначені задачі можуть мати досить великі розміри, для їхнього розв'язування можна успішно застосовувати

стандартне програмне забезпечення для задач лінійного і квадратичного програмування. Так, наприклад, задачі лінійного програмування із сотнями тисяч змінних розв'язуються всього за декілька хвилин на сучасних персональних комп'ютерах.

Отже, знаходження параметрів 3D-структур при $m \approx 500$ і $n \approx 500$ виглядає досить реалістичним за допомогою сучасних програм для розв'язування задач лінійного програмування. Для задачі квадратичного програмування розміри m і n будуть трохи меншими.

Менш чутливими до розмірів 3D-структур будуть методи МНК і МНМ, які використовують оптимізаційні задачі, отримані безпосереднім спрощенням задач (3.2.5) і (3.2.5) – (3.2.6).

МНК для знаходження параметрів регулярної 3D-структури пов'язаний із розв'язуванням такої задачі: знайти

$$\left(f_2^*\right)^2 = \left(f_2\left(x^*, y^*\right)\right)^2 =$$

$$= \min_{x \in R^m,\ y \in R^n} \left\{ \left(f_2(x,y)\right)^2 = \sum_{i=1}^{m} \sum_{j=1}^{n} \left(a_{ij} - x_i - y_j\right)^2 \right\}, \qquad (3.2.14)$$

а для знаходження параметрів базисної регулярної 3D-структури – з розв'язуванням задачі: знайти

$$\left(f_2^*\right)^2 = \left(f_2\left(x^*, y^*\right)\right)^2 =$$

$$= \min_{x \in R^n,\ y \in R^n} \left\{ \sum_{i=1}^{n} \sum_{j=1}^{m} \left(a_{ij} - x_i - y_j\right)^2 + \left(\sum_{i=1}^{n} x_i - \sum_{j=1}^{m} y_j \right)^2 \right\}. \quad (3.2.15)$$

Задачі (3.2.14) і (3.2.15) є задачами мінімізації опуклої квадратичної функції із $N = m + n$ змінними, з тією відмінністю, що перша задача має багато розв'язків, а друга – єдиний розв'язок.

Задачі (3.2.14) і (3.2.15) відрізняються від тих негладких задач, які безпосередньо випливають із формулювань (3.2.5) і (3.2.5) – (3.2.6). У них з L_2-норми вилучена операція добування кореня квадратного, й зроблене це для того, щоб мати справу з мінімізацією квадратичних функцій.

Крім того, для задачі (3.2.15) штраф за порушення рівності (3.2.6) обраний рівним одиниці. Це пояснюється тим, що в цьому випадку знаходження мінімуму квадратичної функції пов'язане з розв'язуванням «блочно-діагональної» системи лінійних рівнянь

$$\begin{vmatrix} nI_m + e_m e_m^T & 0 \\ 0 & mI_n + e_n e_n^T \end{vmatrix} \begin{vmatrix} x^{**} \\ y^{**} \end{vmatrix} = \begin{vmatrix} \sum_{j=1}^{n} a_{ij},\ i = \overline{1,m} \\ \sum_{i=1}^{m} a_{ij},\ j = \overline{1,n} \end{vmatrix}, \qquad (3.2.16)$$

де I_n – одинична $n \times n$-матриця, I_m – одинична $m \times m$-матриця.

З (3.2.16) одержуємо аналітичний розв'язок задачі (3.2.15), який має такий вигляд

$$\begin{aligned} x_i^{**} &= \frac{1}{n}\sum_{j=1}^{n} a_{ij} - \frac{1}{n(n+m)}\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij},\ i = \overline{1,m}, \\ y_j^{**} &= \frac{1}{m}\sum_{i=1}^{m} a_{ij} - \frac{1}{m(n+m)}\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij},\ j = \overline{1,n}. \end{aligned} \qquad (3.2.17)$$

Формули (3.2.17) пояснюють, чому МНК не може відновити регулярну 3D-структуру з одним елементарним дефектом. Дійсно, якщо для якоїсь пари індексів i, j маємо $a_{ij} = u_i^* + v_j^* + \Delta$, де $\Delta \neq 0$, то з формул (3.2.17) випливає, що $x_i^{**} \neq u_i^*$ для всіх $i = \overline{1,n}$ і $y_j^{**} \neq v_j^*$ для всіх $j = \overline{1,m}$.

Крім того, для єдиного елементарного дефекту $a_{ij} = u_i^* + v_j^* + \Delta$ мають місце формули

$$\begin{aligned} \left\| x^{**} - u^* \right\| &= \sqrt{\frac{n+m-2}{n^2(n+m)} + \frac{m}{n^2(n+m)^2}}\,|\Delta| \neq 0, \\ \left\| y^{**} - v^* \right\| &= \sqrt{\frac{n+m-2}{m^2(n+m)} + \frac{n}{m^2(n+m)^2}}\,|\Delta| \neq 0. \end{aligned} \qquad (3.2.18)$$

Якщо формули (3.2.18) застосувати для $\Delta = 1,\ m = 3$ і $n = 5$, то одержимо відхилення $\left\| u^* - x_p^{**} \right\| = 0.17854$ і $\left\| v^* - y_p^{**} \right\| = 0.30334$ з останнього рядка табл. 3.2.1, які при $p = 2$ були отримані за допомогою *r*-алгоритму для регулярної 3D-структури з одним дефектом $a_{23} = a_{23}^* + 1$.

Задачі (3.2.14) і (3.2.15) можуть служити тестовими прикладами для перевірки збіжності алгоритмів мінімізації гладких опуклих функцій. Аналітичний розв'язок (3.2.18) у комбінації з формулами леми 3.2.2 дозволяє оцінити відхилення знайденого розв'язку від точного. При цьому, якщо для задачі (3.2.15) застосовні алгоритми ньютонівського типу, то для задачі (3.2.14) вони незастосовні, тому що матриця квадратичної форми вироджена, її ранг рівний $m + n - 1$ і на одиницю менший, ніж кількість змінних $N = m + n$.

Але зате задача (3.2.14) є зручною для градієнтних методів, зокрема для граничних варіантів *r*-алгоритмів, які сходяться не більше, ніж за N ітерацій (див. теорему 1 у роботі [6]). Для неї, враховуючи що множина розв'язків є неоднозначною, легше знайти одну точку із множини оптимальних точок, ніж єдину оптимальну точку в задачі (3.2.15).

Тому алгоритми, що розв’язують обидві задачі із близькими витратами для однієї й тієї ж точності, будуть кращими, ніж алгоритми, які добре розв’язують одну задачу і погано – іншу. До того, задачу (3.2.15) можна ускладнити, помноживши останній член суми на деякий штраф $M > 0,$ керування яким дозволяє змінювати ступінь витягнутості квадратичних функцій.

На відміну від МНК, оптимізаційні задачі для МНМ випливають із (3.2.5) і (3.2.5) – (3.2.6) без усяких ускладнень. Так, МНМ для знаходження параметрів регулярної 3D-структури пов'язаний із розв’язуванням такої задачі:

знайти

$$f_1^* = f_1\left(x^*, y^*\right) = \min_{x \in R^m,\ y \in R^n} \left\{ f_1(x, y) = \sum_{i=1}^{m} \sum_{j=1}^{n} \left| a_{ij} - x_i - y_j \right| \right\}, \quad (3.2.19)$$

а для знаходження параметрів базисної регулярної 3D-структури – з розв’язуванням задачі:

знайти

$$f_1^* = f_1\left(x^*, y^*\right) = \min_{x \in R^m,\ y \in R^n} \left\{ \sum_{i=1}^{m} \sum_{j=1}^{n} \left| a_{ij} - x_i - y_j \right| + \left| \sum_{i=1}^{m} x_i - \sum_{j=1}^{n} y_j \right| \right\}. \quad (3.2.20)$$

Задачі (3.2.19) і (3.2.20) є задачами мінімізації опуклої кусково-лінійної функції $N = m + n$ змінних, де перша задача має багато розв'язків, а друга – один розв'язок. Для розв'язку негладких задач (3.2.19) і (3.2.20) можна використовувати як субградієнтні методи мінімізації негладких опуклих функцій [3, 4], так і спеціальні ітеративні методи, наприклад, метод варіаційно-зважених квадратичних наближень [7].

Робастність МНМ для 3D-структур з дефектами підтверджена обчислювальними експериментами у попередньому розділі. Це означає, що, якщо 3D-структура є регулярною й до неї додано одну або кілька областей з дефектами, то параметри регулярної 3D-структури більш точно визначаються за допомогою МНМ, ніж за допомогою МНК.

Це підтверджується рис. 3.2.4, де проілюстровані результати роботи обох методів для автоматичного визначення нерегулярності в 3D-структурах. Це зроблене для двох зображень, які є аналогічними тим, що одержуються методами лазерної інтерферометрії (рис. 3.2.4, у центрі).

Верхнє зображення містить одну область із дефектами, а нижнє – дві області з дефектами. Перша дефектна область із центром у точці з координатами $(i, j) = (55, 74)$ зі значним відхиленням від регулярної структури легко ідентифікується оператором неруйнуючого контролю якості. Друга дефектна область із центром у точці з координатами $(i, j) = (37, 37)$ може бути пропущена оператором через низьку контрастність зображення.

На рис. 3.2.4 (ліворуч і праворуч) наведені абсолютні різниці між коефіцієнтами матриці для 3D-структур з дефектами й коефіцієнтами

матриць регулярних 3D-структур, знайдених за допомогою МНК і МНМ. З цього рисунка випливає, що МНМ однозначно визначає координати дефектів, на відміну від МНК, який помилково виділяє ділянки уздовж стовпців і рядків кожного дефекту.

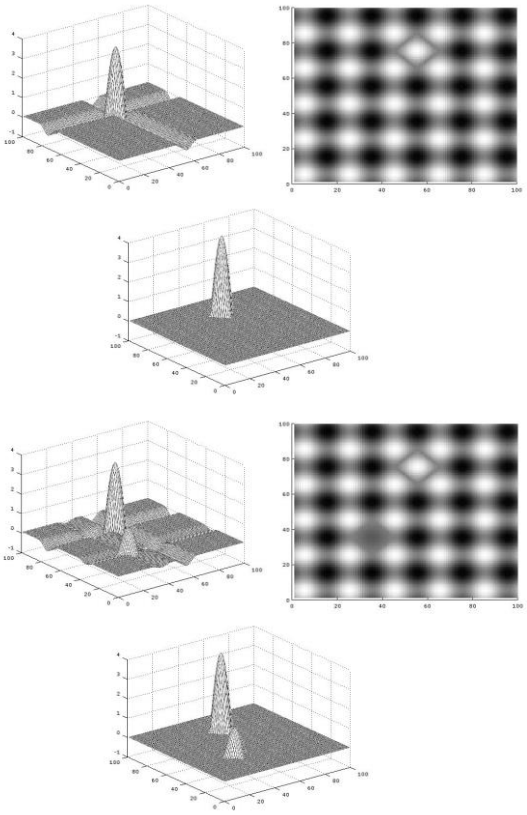


Рис. 3.2.4. Результати МНК (ліворуч) і МНМ (праворуч) для регулярних 3D-структур з дефектами

На основі Octave-Функції ralgb5 [5], однієї із сучасних реалізацій r-алгоритму Шора, розроблено програми мовою Octave для МНМ, де розв’язується задача (3.2.14), і для МНК, де розв’язується задача

(3.2.19). Для регулярної 3D-структури, де $n = 600$ і $m = 400$, часові та інші затрати обох програм наведені в табл. 3.2.2.

Обчислення проводилися на комп'ютері Pentium 3 Ghz у системі Windows 7/32 використовуючи GNU Octave версії 3.6.4.

Табл. 3.2.2. Затрати програм для МНМ і МНК при знаходженні параметрів регулярної 3D-структури, $m = 400$ і $n = 600$

№	МНК			МНМ		
п/п	itn	Nfg	time	itn	nfg	time
1	457	700	10.58	411	502	11.76
2	494	785	11.59	413	505	11.79
3	426	647	9.77	412	503	11.77
4	453	702	10.50	412	502	11.77
5	488	764	11.34	410	499	11.69

У табл. 3.2.2 для п'яти тестових прикладів $a_{ij} = u_i + v_j$, де компоненти векторів $u \in R^{400}$ і $v \in R^{600}$ генерувалися датчиком випадкових чисел у діапазоні від 0 до 10, наведено: itn – кількість ітерацій, nfg – кількість обчислень значення функції і її субградієнта, time – час виконання програми у секундах. З табл. 3.2.2 бачимо, що для розв'язання задач за допомогою МНК і МНМ досить близько 10 секунд, що означає, що розроблені програми можна використовувати в діалоговому режимі для знаходження параметрів регулярних зображень.

3.2.5 Про нерівності для суми квадратів двох наборів чисел

У підрозділі 3.2.3 при описі базисних регулярних 3D-структур (означення 3.2.2) неявно використовувалася нерівність, яка пов'язана з формулами (3.2.1) і (3.2.2) для мінімізації одномірної квадратичної функції $f(t)$. Оскільки ця нерівність може мати самостійний інтерес, то нижче викладемо її, розраховуючи у загальному випадку.

Теорема 3.2.2. Для n чисел $a_1,...,a_n$ і m чисел $b_1,...,b_m$ завжди справедлива нерівність

$$\sum_{i=1}^{n} a_i^2 + \sum_{j=1}^{m} b_j^2 \geq \frac{\left(\sum_{i=1}^{n} a_i - \sum_{j=1}^{m} b_j\right)^2}{n+m}, \qquad (3.2.21)$$

яка як рівність виконується тільки тоді, коли $a_1 = ... = a_n = c$ і $b_1 = ... = b_m = -c$, де c – довільна константа.

Доведення. Покажемо справедливість нерівності

$$\sum_{i=1}^{n} a_i^2 + \sum_{j=1}^{m} b_j^2 - \frac{\left(\sum_{i=1}^{n} a_i - \sum_{j=1}^{m} b_j\right)^2}{n+m} \geq 0, \qquad (3.2.22)$$

яка рівносильна нерівності (3.2.21). Ліву частину нерівності (3.2.22) можна записати у вигляді

$$\sum_{i=1}^{n} a_i^2 + \sum_{j=1}^{m} b_j^2 - \frac{\left(\sum_{i=1}^{n} a_i - \sum_{j=1}^{m} b_j\right)^2}{n+m} =$$

$$= \sum_{i=1}^{n} a_i^2 + \sum_{j=1}^{m} b_j^2 - \frac{2\left(\sum_{i=1}^{n} a_i - \sum_{j=1}^{m} b_j\right)^2}{n+m} + \frac{(m+n)\left(\sum_{i=1}^{n} a_i - \sum_{j=1}^{m} b_j\right)^2}{(n+m)^2} =$$

$$= \sum_{i=1}^{n} a_i^2 - 2\sum_{i=1}^{n} a_i \frac{\left(\sum_{i=1}^{n} a_i - \sum_{j=1}^{m} b_j\right)}{n+m} + \frac{n\left(\sum_{i=1}^{n} a_i - \sum_{j=1}^{m} b_j\right)^2}{(n+m)^2} +$$

$$+ \sum_{j=1}^{m} b_j^2 + 2\sum_{j=1}^{m} b_j \frac{\left(\sum_{i=1}^{n} a_i - \sum_{j=1}^{m} b_j\right)}{n+m} + \frac{m\left(\sum_{i=1}^{n} a_i - \sum_{j=1}^{m} b_j\right)^2}{(n+m)^2} =$$

$$= \sum_{i=1}^{n} \left(a_i - \frac{\left(\sum_{i=1}^{n} a_i - \sum_{j=1}^{m} b_j \right)}{n+m} \right)^2 + \sum_{j=1}^{m} \left(b_j + \frac{\left(\sum_{i=1}^{n} a_i - \sum_{j=1}^{m} b_j \right)}{n+m} \right)^2 .$$

Отже, нерівність (3.2.22) можна переписати в еквівалентній формі

$$\sum_{i=1}^{n} \left(a_i - \frac{\left(\sum_{i=1}^{n} a_i - \sum_{j=1}^{m} b_j \right)}{n+m} \right)^2 + \sum_{j=1}^{m} \left(b_j + \frac{\left(\sum_{i=1}^{n} a_i - \sum_{j=1}^{m} b_j \right)}{n+m} \right)^2 \geq 0 , \quad (3.2.23)$$

для якої є очевидним, що вона буде виконуватися для довільних послідовностей чисел $a_1,...,a_n$ і $b_1,...,b_m$. Більше того, з нерівності (3.2.23) видно, що вона перетворюється у рівність, якщо

$$a_1 = \ldots = a_n = \frac{\left(\sum_{i=1}^{n} a_i - \sum_{j=1}^{m} b_j \right)}{n+m}, \quad b_1 = \ldots = b_m = -\frac{\left(\sum_{i=1}^{n} a_i - \sum_{j=1}^{m} b_j \right)}{n+m},$$

що можливо тільки тоді, коли $a_1 = ... = a_n = c$ і $b_1 = ... = b_m = -c$, де c – довільна константа.

Доведення теореми завершено.

Якщо у теоремі 3.2.2 замість послідовності чисел $b_1,...,b_m$ використовувати послідовність чисел $-b_1,...,-b_m$, то одержимо справедливість ще однієї нерівності, яка, мабуть, навіть красивіша, ніж нерівність (3.2.21).

Теорема 3.2.3. Для n чисел $a_1,...,a_n$ і m чисел $b_1,...,b_m$ завжди справедлива нерівність

$$\sum_{i=1}^{n} a_i^2 + \sum_{j=1}^{m} b_j^2 \geq \frac{\left(\sum_{i=1}^{n} a_i + \sum_{j=1}^{m} b_j \right)^2}{n+m}, \quad (3.2.24)$$

яка як рівність виконується тільки тоді, коли $a_1 = ... = a_n = b_1 = ... = b_m$.

Теорему 3.2.3 легко довести за тією ж самою схемою, за якою доведена теорема 3.2.2.

З теорем 3.2.2 і 3.2.3 випливає низка відомих нерівностей. Так, наприклад, якщо $m = n = 1$, то одержуємо добре відомі «шкільні» нерівності

$$a^2 + b^2 \geq \frac{(a-b)^2}{2} \quad \text{та} \quad a^2 + b^2 \geq \frac{(a+b)^2}{2}.$$

Якщо вибрати $n = m$ та однакові послідовності чисел, тобто $b_1 = a_1, ..., b_n = a_n$, то з нерівності (3.2.24) теореми 3.2.3 одержимо нерівність

$$a_1^2 + \ldots + a_n^2 \geq \frac{(a_1 + \ldots + a_n)^2}{n}, \qquad (3.2.25)$$

яка виконується як рівність тільки тоді, коли $a_1 = ... = a_n$. Цю ж нерівність можна одержати й з теореми 3.2.2, якщо вибрати $n = m$ і $b_1 = -a_1, ..., b_n = -a_n$.

Задача «Доведіть нерівність (3.2.25)» включена до книги [8] – задача під номером 8.10 на стор. 97. У книзі дано такий розв'язок задачі 8.10: «Нерівність (3.2.25) є частковим випадком нерівності Коші

$$\left(\sum_{i=1}^{n} a_i b_i\right)^2 \leq \left(\sum_{i=1}^{n} a_i^2\right)^2 \left(\sum_{i=1}^{n} b_i^2\right)^2, \qquad (3.2.26)$$

у якій досить покласти $b_1 = ... = b_n = 1$».

Однак, елегантнішим буде доведення, яке легко провести за тією ж схемою, що й доведення теореми 3.2.2. Для цього досить нерівність (3.2.25) замінити на еквівалентну нерівність

$$a_1^2 + \ldots + a_n^2 - \frac{(a_1 + \ldots + a_n)^2}{n} \geq 0, \qquad (3.2.27)$$

для якої ліву частину можна записати таким чином

$$a_1^2 + \ldots + a_n^2 - \frac{(a_1 + \ldots + a_n)^2}{n} = a_1^2 + \ldots + a_n^2 - \frac{2(a_1 + \ldots + a_n)^2}{n} +$$

$$+\frac{n(a_1+\ldots+a_n)^2}{n^2}=a_1^2-2a_1\frac{(a_1+\ldots+a_n)}{n}+\frac{(a_1+\ldots+a_n)^2}{n^2}+\ldots$$

$$\ldots+a_n^2-2a_n\frac{(a_1+\ldots+a_n)}{n}+\frac{(a_1+\ldots+a_n)^2}{n^2}=$$

$$=\left(a_1-\frac{(a_1+\ldots+a_n)}{n}\right)^2+\ldots+\left(a_n-\frac{(a_1+\ldots+a_n)}{n}\right)^2.$$

Отже, нерівність (3.2.17) можна переписати в еквівалентній формі

$$\left(a_1-\frac{(a_1+\ldots+a_n)}{n}\right)^2+\ldots+\left(a_n-\frac{(a_1+\ldots+a_n)}{n}\right)^2\geq 0\,, \quad (3.2.28)$$

звідки є очевидним, що нерівність (3.2.28) виконується для n довільних чисел $a_1,\ldots,a_n$.

Аналогічне доведення можна провести також для нерівності (3.2.16) – нерівності Коші – Буняковського – Шварца. Для цього досить нерівність (3.2.16) замінити на еквівалентну нерівність

$$\left(\sum_{i=1}^n a_i^2\right)^2-\frac{\left(\sum_{i=1}^n a_ib_i\right)^2}{\left(\sum_{i=1}^n b_i^2\right)^2}\geq 0, \quad (3.2.29)$$

у припущенні $\sum_{i=1}^n b_i^2>0$. Ліву частину нерівності (3.2.29) можна записати таким чином:

$$\left(\sum_{i=1}^n a_i^2\right)^2-\frac{\left(\sum_{i=1}^n a_ib_i\right)^2}{\left(\sum_{i=1}^n b_i^2\right)^2}=\left(\sum_{i=1}^n a_i^2\right)^2-\frac{2\left(\sum_{i=1}^n a_ib_i\right)\left(\sum_{i=1}^n a_ib_i\right)}{\left(\sum_{i=1}^n b_i^2\right)^2}+$$

$$+\frac{\left(\sum_{i=1}^{n} b_i^2\right)^2 \left(\sum_{i=1}^{n} a_i b_i\right)^2}{\left(\sum_{i=1}^{n} b_i^2\right)^4} = \sum_{i=1}^{n} \left(a_i^2 - \frac{2a_i b_i \left(\sum_{i=1}^{n} a_i b_i\right)}{\left(\sum_{i=1}^{n} b_i^2\right)^2} + \frac{b_i^2 \left(\sum_{i=1}^{n} a_i b_i\right)^2}{\left(\sum_{i=1}^{n} b_i^2\right)^4} \right) =$$

$$= \sum_{i=1}^{n} \left(a_i^2 - \frac{\left(\sum_{i=1}^{n} a_i b_i\right)}{\left(\sum_{i=1}^{n} b_i^2\right)^2} b_i \right)^2 .$$

Отже, нерівність (3.2.29) можна переписати в еквівалентній формі

$$\sum_{i=1}^{n} \left(a_i^2 - \frac{\left(\sum_{i=1}^{n} a_i b_i\right)}{\left(\sum_{i=1}^{n} b_i^2\right)^2} b_i \right)^2 \geq 0 , \qquad (3.2.30)$$

звідки є очевидним, що нерівність (3.2.30) буде виконуватися для довільних n чисел $a_1,...,a_n$ і n чисел $b_1,...,b_n$.

3.2.6 Висновки

У розділі, матеріал якого базується на статті [9], розглянуто регулярні та базисні регулярні 3D-структури, для них сформульовано оптимізаційні задачі знаходження найкращих по L_p-нормі параметрів і досліджено методи розв'язування розглянутих задач. Особливу увагу приділено окремим випадкам методів – методу найменших квадратів і методу найменших модулів.

Показано, що при відновленні параметрів 3D-структур з дефектами метод найменших модулів є стійкішим, ніж метод найменших квадратів. Наведено результати обчислювальних експериментів для програмних реалізацій методів на основі r-

алгоритму, які дозволяють оцінити їхні часові затрати при знаходженні параметрів регулярних 3D-структур великих розмірів (400 рядків і 600 стовпців).

Регулярні зображення з областями дефектів є характерними при неруйнуючому контролі якості тонкостінних багатошарових композиційних матеріалів за допомогою методів лазерної інтерферометрії, таких, як метод голографічної інтерферометрії, метод спекл-інтерферометрії та метод ширографії [1].

Описані в розділі методи для пошуку дефектних ділянок дозволяють автоматизувати процес визначення місця розташування дефектів у відповідальних елементах конструкцій і знизити вплив людського фактору при неруйнуючому контролі якості. Оптимізаційні задачі для зображень із 400 пікселями по вертикалі та 600 пікселями по горизонталі можна розв’язувати в діалоговому режимі, тому що для цього потрібно близько 10 секунд на сучасних персональних ЕОМ з використанням GNU Octave версії 3.6.4.

Розроблені Octave-Програми легко переписати мовами Фортран і Сі, використовуючи бібліотеку базових підпрограм лінійної алгебри BLAS (Basic Linear Algebra Subprograms) або бібліотеку математичних прикладних програм Intelr Math Kernel Library (Intelr MKL), які оптимізовані під сучасні обчислювальні машини.

Це може дозволити значно прискорити методи для розв'язування великих задач (з тисячею і більше змінними), наприклад, за рахунок використання обчислювальних потужностей графічного процесора, які в рази перевищують обчислювальні потужності класичних процесорів.

Так, істотне прискорення можна одержати, використовуючи для розрахунків технологію CUDA на графічних прискорювачах. Це підтверджує гібридна реалізація r-алгоритму [10], де скорочення часу, яке досягається при розв'язувані задач із 1000–8000 змінними, варіюється від 14 до 18 разів.

По аналогії з алгоритмами для регулярних структур, наведеними у статті [11], розроблено метод пошуку дефектів у періодичних структурах за допомогою методу порівняння експериментального та ідеалізованого образів.

Показана ефективність «урізаної» L_2-норми для відновлення параметрів ідеалізованої моделі. Описана програмна реалізація методу та наведені результати обчислювальних експериментів для зображень, отриманих методом ширографії.

Список літератури

1. Lobanov L.M., Pivtorak V.A., Kyjanets I.V., Savitsky V.V., Tkachuk G.I. Express control of quality and stressed state of welded structures using method of electron shearography and speckle-interferometry. The Paton Welding Journal. August, 2005. P. 35–40.
2. Стецюк П.И., Савицкий В.В. О робастности метода наименьших модулей для поиска дефектов в регулярных 3D-структурах. Матеріали Дев'ятнадцятого Міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування» присвяченого пам'яті д.ф.-м.н., професора Петренюка Анатолія Яковича (Кропивницький, 7-8 квітня 2017 року). За ред. Г.П. Донця. Кропивницький. 2017. С. 127–132.
3. Шор Н.З. Методы минимизации недифференцируемых функций и их приложения. Киев: Наук. думка. 1979. 200 с.
4. Стецюк П.И. Методы эллипсоидов и r-алгоритмы. Кишинэу: Эврика. 2014. 488 с.
5. Стецюк П.И. Субградиентные методы ralgb5 и ralgb4 для минимизации овражных выпуклых функций. *Вычислительные технологии.* 2017. Т. 22. № 2. С. 127–149.
6. Стецюк П.И. Теория и программные реализации r-алгоритмов Шора. *Кибернетика и системный анализ.* 2017. № 5. С. 43–57.
7. Мудров В.И., Кушко В.Л. Метод наименьших модулей. М.: Знание. 1971. 64 с.
8. Прасолов В.В. Задачи по алгебре, арифметике и анализу: Учебное пособие. М.: МЦНМО. 2007. 608 с.

9. Стецюк П.И., Савицкий В.В. О поиске дефектов в регулярных 3D-структурах. *Проблемы управления и информатики.* 2018. № 2. С. 33–48.
10. Стецюк П.І., Хіміч О.М., Сидорук В.А. Реалізація r-алгоритму на графічних процесорах. *Комп'ютерна математика*. 2016. № 2. С. 100–109.
11. Жидков В.А. Определение дефектов в периодических структурах. *Компьютерная математика.* Киев: Ин-т кибернетики им. В.М. Глушкова НАН Украины. 2017. № 2. С. 21–29.

3.3 ПОБУДОВА S-ПОДІБНОЇ ПАРАМЕТРИЧНОЇ КРИВОЇ ТА ЇЇ ЗАСТОСУВАННЯ ДЛЯ ПРОЕКТУВАННЯ ЗОВНІШНЬОГО КОНТУРА СОПЛА

П. І. Стецюк, О. М. Хом'як, В. О. Жидков

Анотація. Описано математичну модель, алгоритм та програмне забезпечення для задачі побудови *S*-подібної кривої, яка проходить через дві задані точки із заданими кутами нахилу дотичних у них та забезпечує заданий кут нахилу дотичної в точці із заданою абсцисою. Для керування точкою перегину *S*-подібної кривої з квадратичним законом розподілу кривини в натуральній параметризації використовується кут нахилу дотичної в точці із заданою абсцисою. Алгоритм базується на модифікації градієнтного методу з розтягом простору в напрямі різниці двох послідовних узагальнених градієнтів. Наведено опис програмної реалізації та обчислювальних експериментів, які показали ефективність розробленого алгоритму для проектування зовнішнього контуру сопла типу Франкля.

Abstract. A mathematical model, algorithm and software are described for the problem of S-shaped curve constructing that passes through two given points with given angles of inclination of tangents at them and provides a given angle of inclination of tangent at a point with a given abscissa. To control the inflection point of an S-shaped curve with quadratic law of curvature distribution in natural parameterization, the angle of inclination of tangent at a point with a given abscissa is used. The algorithm is based on a gradient method modification with space dilation in the direction of the difference of two consecutive generalized gradients. A description of the software implementation and computational experiments showing effectiveness of the developed algorithm for designing the external contour of a Frankl-type nozzle is given.

3.3.1 Вступ

У даному параграфі, матеріал якого базується на роботі [1], описано математичну модель, алгоритм та програмне забезпечення для задачі побудови *S*-подібної кривої, яка проходить через дві задані точки із заданими кутами нахилу дотичних у них та забезпечує заданий кут нахилу дотичної в точці із заданою абсцисою. Для керування точкою перегину *S*-подібної кривої з квадратичним законом розподілу кривини в натуральній параметризації використовується кут нахилу дотичної в точці із заданою абсцисою. Алгоритм основано на модифікації методу з розтягом простору в напрямі різниці двох послідовних узагальнених градієнтів. Обчислювальні експерименти показали ефективність розробленого алгоритму та його програмної реалізації для проектування фрагментів зовнішнього контуру сопла Франкля (див. рис. 3.3.1).

Алгоритм використовується для побудови зовнішнього контуру сопла типу Франкля за допомогою двохланкових S-подібних кривих, отриманих з використанням алгоритмів натуральної параметризації та квадратичного закону розподілу кривини. Зовнішній контур сопла типу Франкля складається з двох S-подібних кривих: перша крива – для дозвукової частини (де сопло звужується) та друга крива – для надзвукової частини (де сопло розширюється). Керування точками перегину цих кривих дає можливість вибирати необхідні форми для фрагментів зовнішнього контуру дозвукової та надзвукової частин сопла.

3.3.2 Геометрична модель задачі

Вихідними даними для побудови контуру у вигляді плоскої S-подібної кривої є координати точок (x_1, y_1), (x_2, y_2) та кути нахилу дотичних у них (φ_1, φ_2), а також кут φ_p – кут нахилу дотичної в точці (x_p, y_p) та її абсциса x_p, які будуть використовуватися для керування точкою перегину S-подібної кривої. Для надзвукового фрагменту

контуру сопла Франкля задано точки та кути нахилу в них наведено на рис. 3.3.1.

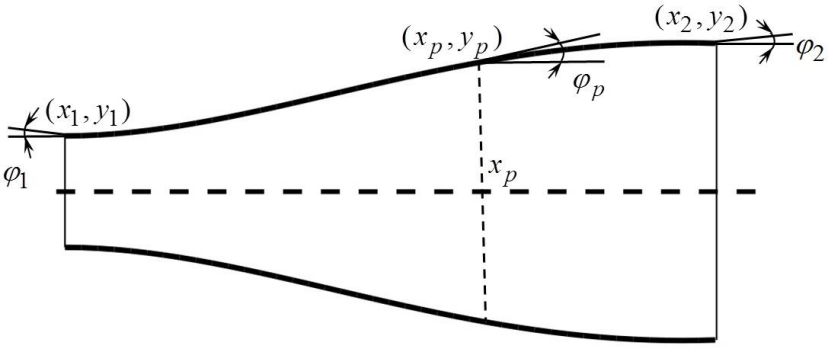


Рис. 3.3.1. Вихідні дані для зовнішнього контуру сопла Франкля у надзвуковій частині

Задача полягає у наступному. Потрібно так з'єднати точки (x_1, y_1) та (x_2, y_2) кривою лінією в натуральній параметризації, де кривина $k(s) = as^2 + bs + c$ є квадратичною функцією від s – довжини кривої, щоб забезпечити в точках (x_1, y_1) та (x_2, y_2) задані значення кутів нахилу дотичних φ_1 та φ_2, а в точці з абсцисою x_p, для якої $x_1 < x_p < x_2$, забезпечити кут рівний φ_p. Кути φ_1, φ_2 та φ_p вимірюються в радіанах.

Якщо контур представляється у вигляді *S*-подібної кривої, то квадратична кривина $k(s) = as^2 + bs + c$ забезпечує єдиність розв'язку задачі. Задаючи значення абсциси x_p та кут нахилу дотичної φ_p, отримуємо можливість керувати точкою перегину отриманої *S*-подібної кривої, що дозволяє автономно моделювати той чи інший вигляд фрагментів зовнішнього контуру сопла Франкля у дозвуковій та надзвуковій його частинах. Для дозвукової частини сопла фрагмент

зовнішнього контуру має вигляд, який є «дзеркальним» відображенням фрагменту зовнішнього контуру у надзвуковій частині. Для надзвукової частини сопла абсциса x_p буде розміщуватися ближче до її закінчення, а для дозвукової частини сопла абсциса x_p буде розміщуватися ближче до її початку.

Для кривої $x(s)$, $y(s)$, $s>0$ в натуральній параметризації кут нахилу дотичної $\varphi(s)$ в точці $(x(s),y(s))$ визначається за формулою

$$\varphi(s)=\varphi(0)+\int_0^s k(s)ds=\varphi(0)+\frac{as^3}{3}+\frac{bs^2}{2}+cs, \qquad (3.3.1)$$

а координати $x(s)$ та $y(s)$ – за формулами

$$x(s)=x(0)+\int_0^s \cos\varphi(s)ds, \quad y(s)=y(0)+\int_0^s \sin\varphi(s)ds. \qquad (3.3.2)$$

Тут кривина $k(s)=as^2+bs+c$ задається квадратичною функцією від s – довжини кривої, де a, b, c – задані коефіцієнти.

3.3.3 Система нелінійних інтегральних рівнянь. Її властивості

Нехай S – довжина кривої від точки (x_1,y_1) до точки (x_2,y_2), а s_p – довжина кривої від точки (x_1,y_1) до точки (x_p,y_p). Знаходженню параметрів кривини a, b, c та довжин S, s_p відповідає система з п'яти нелінійних рівнянь [2]:

$$x_2=x_1+\int_0^S \cos\left(\varphi_1+\frac{as^3}{3}+\frac{bs^2}{2}+cs\right)ds, \qquad (3.3.3)$$

$$y_2=y_1+\int_0^S \sin\left(\varphi_1+\frac{as^3}{3}+\frac{bs^2}{2}+cs\right)ds, \qquad (3.3.4)$$

$$\varphi_2=\varphi_1+\frac{aS^3}{3}+\frac{bS^2}{2}+cS, \qquad (3.3.5)$$

$$x_p = x_1 + \int_0^{s_p} \cos\left(\varphi_1 + \frac{as^3}{3} + \frac{bs^2}{2} + cs\right) ds\,, \qquad (3.3.6)$$

$$\varphi_p = \varphi_1 + \frac{as_p^3}{3} + \frac{bs_p^2}{2} + cs_p\,. \qquad (3.3.7)$$

Система (3.3.3) – (3.3.7) має п'ять невідомих: a, b, c – три коефіцієнти квадратичної функції, S – загальна довжина кривої, s_p – довжина ділянки кривої до точки з відомою абсцисою. Система включає п'ять нелінійних рівнянь, серед яких рівняння (3.3.3), (3.3.4) та (3.3.6) є інтегральними та залежать від невідомих параметрів підінтегральних функцій та невідомих верхніх границь для визначеного інтегралу. Інтегральні рівняння (3.3.3) та (3.3.4) зв'язують між собою точки (x_1, y_1) та (x_2, y_2) за формулами (3.3.2). Нелінійне рівняння (3.3.5) за формулою (3.3.1) забезпечує потрібний кут φ_2 в точці (x_2, y_2), який визначається за заданим кутом φ_1 в точці (x_1, y_1). Інтегральне рівняння (3.3.6) зв'язує між собою координати x_1 та x_p за першою формулою із (3.3.2), а нелінійне рівняння (3.3.7) забезпечує кут рівний φ_p в точці з абсцисою x_p за формулою (3.3.1).

Кут нахилу дотичної φ_p в точці з відомою абсцисою x_p будемо використовувати для керування точкою перегину *S*-подібної кривої, в якій кривина є рівною нулю. Зауважимо, що ордината y_p не використовується як невідома змінна. Її значення обчислюється за формулою $y_p = y_1 + \int_0^{s_p} \sin\left(\varphi_1 + \frac{as^3}{3} + \frac{bs^2}{2} + cs\right) ds$, де s_p – знайдена довжина ділянки кривої до точки з відомою абсцисою.

Виділимо дві властивості системи (3.3.3) – (3.3.7), які потрібно враховувати при знаходженні її розв'язків, пов'язаних з проектуванням зовнішнього контуру сопла Франкля у вигляді двохланкових S-подібних кривих.

Перша властивість: у загальному випадку система (3.3.3) – (3.3.7) має багато розв'язків. Так, наприклад, у табл. 3.3.1 наведено чотири розв'язки для наступних вихідних даних:

$$x_1 = 0,\ y_1 = 2.46,\ \varphi_1 = 0; \quad x_2 = 1,\ y_2 = 2.75,\ \varphi_2 = 0;$$
$$x_p = 0.7,\ \varphi_p = 0.209440 = 12^{\circ}. \tag{3.3.8}$$

Графіки кривих для цих розв'язків наведені на рис. 3.3.2 (криві для першого і другого розв'язків) та на рис. 3.3.3 (криві для третього та четвертого розв'язків).

Табл. 3.3.1. Чотири розв'язки системи (3.3.3) – (3.3.7) для вихідних даних (3.3.8)

	1	2	3	4
a^*	7.92822	-7.03716	8.16618	-6.67708
b^*	-11.4065	16.4890	-28.3207	23.8707
c^*	3.07557	-6.19900	18.3901	-15.9001
S^*	1.05562	2.42487	2.69815	2.89409
s_p^*	0.753992	1.15072	2.39725	2.58685

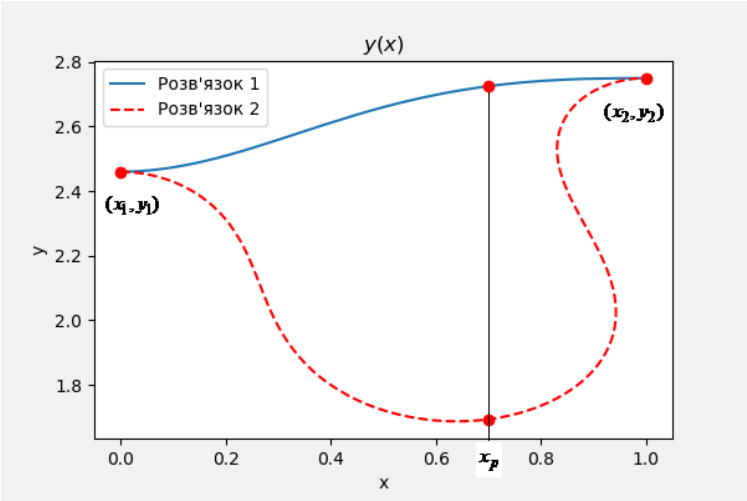


Рис. 3.3.2. Графіки кривих для першого і другого розв'язків із табл. 3.3.1

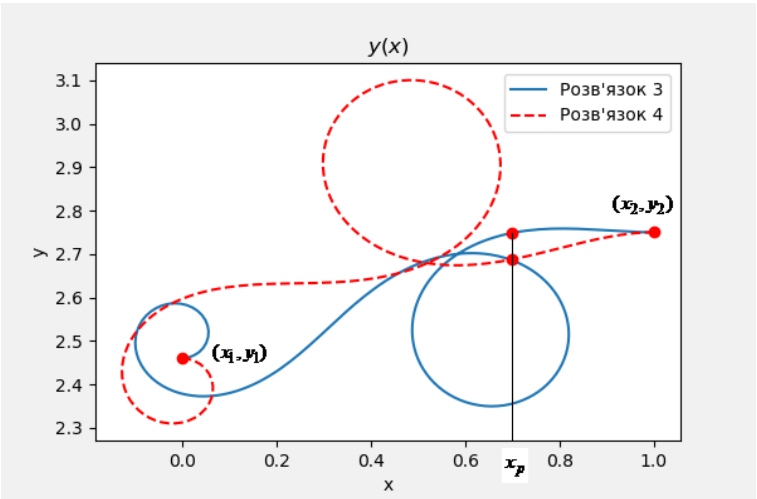


Рис. 3.3.3. Графіки кривих для третього і четвертого розв'язків із табл. 3.3.1

Із рис. 3.3.2 та рис. 3.3.3 видно, що тільки для першого розв'язку крива є *S*-подібною. Вона характеризується наявністю однієї точки перегину, де кривина змінює знак. Її координати $x(s_1)$, $y(s_1)$ визначаються першим розв'язком квадратного рівняння $a^* s^2 + b^* s + c^* = 0$, $s_{1,2} = \dfrac{-b^* \pm \sqrt{(b^*)^2 - 4a^* c^*}}{2a^*}$, $s_1 = 0.35943 \in [0, S^*]$.

Для трьох інших розв'язків це не так. Крива для другого розв'язку включає дві ділянки, кожна із яких є *S*-подібною кривою, та характеризується довжиною $S^* = 2.42487$, яка є значно більшою за довжину $S^* = 1.05562$ для першого розв'язку. Зауважимо, що для другого розв'язку існує ділянка кривої $y(x)$, де ординати визначені неоднозначно. Для кривих, які відповідають третьому та четвертому розв'язкам, існують по дві ділянки, де ординати кривих $y(x)$ визначаються неоднозначно, причому ділянки кривих, які є ближчими до точки (x_2, y_2), мають циклічний характер.

Тому першою умовою для пошуку розв'язку системи (3.3.3) – (3.3.7) є відсікання тих зайвих розв'язків, яким не відповідають *S*-подібні криві. Це легко зробити за допомогою обмеження зверху на довжину кривої. Так, наприклад, якщо до системи рівнянь (3.3.3) – (3.3.7) додати нерівність $S \le 2$, то відсікатимуться другий, третій і четвертий розв'язки із табл. 3.3.1.

Друга властивість: система (3.3.3) – (3.3.7) може бути погано масштабованою (сингулярною). Нехай довжина кривої $S^* = 1000$. Для того, щоб у точці (x_2, y_2) гарантувати близький до нуля кут φ_2, потрібно, щоб для малого ε виконувалася така нерівність

$$-\varepsilon \le \frac{a(1000)^3}{3} + \frac{b(1000)^2}{2} + 1000c \le \varepsilon. \qquad (3.3.9)$$

Нерівність (3.3.9) означає, що компоненти розв'язку системи (3.3.3) – (3.3.7) будуть величинами дуже різних порядків: так a^* буде

мати порядок 10^{-9}, b^* буде мати порядок 10^{-6}, а c^* буде мати порядок 10^{-3}. Це висуває додаткові вимоги для методу розв'язання системи нелінійних рівнянь.

У цьому випадку може допомогти лема [2], яка встановлює зв'язок розв'язків вихідної системи (3.3.3) – (3.3.7) та масштабованої системи, в якій координати точок домножуються на одну і ту ж величину $\mu > 0$.

Лема 3.3.1 ($\mu > 0$). Якщо a^*, b^*, c^*, S^* та s_p^* є розв'язком системи (3.3.3) – (3.3.7), то

$$a^{**} = a^* / \mu^3, b^{**} = b^* / \mu^2, c^{**} = c^* / \mu, S^{**} = \mu S^*, s_p^{**} = \mu s_p^*, \quad (3.3.10)$$

є розв'язком такої системи рівнянь:

$$\mu x_2 = \mu x_1 + \int_0^S \cos\left(\varphi_1 + \frac{as^3}{3} + \frac{bs^2}{2} + cs\right) ds, \quad (3.3.11)$$

$$\mu y_2 = \mu y_1 + \int_0^S \sin\left(\varphi_1 + \frac{as^3}{3} + \frac{bs^2}{2} + cs\right) ds, \quad (3.3.12)$$

$$\varphi_2 = \varphi_1 + \frac{aS^3}{3} + \frac{bS^2}{2} + cS, \quad (3.3.13)$$

$$\mu x_p = \mu x_1 + \int_0^{s_p} \cos\left(\varphi_1 + \frac{as^3}{3} + \frac{bs^2}{2} + cs\right) ds, \quad (3.3.14)$$

$$\varphi_p = \varphi_1 + \frac{as_p^3}{3} + \frac{bs_p^2}{2} + cs_p. \quad (3.3.15)$$

За допомогою леми 3.3.1 можна, використовуючи отриманий розв’язок для добре масштабованої системи (де усі компоненти розв'язку мають один порядок), легко знаходити відповідний йому розв’язок для погано масштабованої (сингулярної) системи. Для цього достатньо знайти розв'язок добре масштабованої системи (3.3.3) – (3.3.7) та перерахувати розв'язок для погано масштабованої системи (3.3.11) – (3.3.15) за формулою (3.3.10).

Зауважимо, що в трохи зміненому вигляді система (3.3.3) – (3.3.7) розглядається у статті [3], де невідома довжина s_p замінюється її

невідомою часткою p у загальній довжині S, тобто $s_p = pS$, $0 < p < 1$. Для знаходження розв'язку системи спочатку з рівнянь (3.3.5) та (3.3.7) виражаються невідомі коефіцієнти a і c через невідомий коефіцієнт b та невідомі довжини S, s_p. Підставляючи їх у рівняння (3.3.3), (3.3.4) та (3.3.6) отримуємо систему трьох інтегральних рівнянь з трьома невідомими b, S, s_p. Використання ітераційних методів для пошуку розв'язку системи трьох рівнянь затруднюється складними виразами для підінтегральних функцій.

Нижче розглянемо метод, який звільнений від вказаного недоліку. У його основу покладено модифікацію *r*-алгоритму [4] для розв'язання спеціальної задачі мінімізації негладкої функції (сума модулів нев'язок системи (3.3.3) – (3.3.7)) при контролі обмежень на довжини S, s_p, щоб гарантувати їх додатні допустимі значення.

3.3.4 Оптимізаційна задача та алгоритм її розв'язання

Розглянемо умовну задачу мінімізації суми модулів функцій нев'язок для рівнянь (3.3.3) – (3.3.7), яка має вигляд: знайти

$$f^* = f\left(a^*, b^*, c^*, S^*, s_p^*\right) =$$

$$= \min_{a,b,c,S,s_p} \left\{ f\left(a,b,c,S,s_p\right) = \sum_{i=1}^{3} \left| f_i\left(a,b,c,S\right)\right| + \sum_{i=4}^{5} \left| f_i\left(a,b,c,s_p\right)\right| \right\} \quad (3.3.16)$$

при обмеженнях

$$S_{\min} \le S \le S_{\max}, \quad (3.3.17)$$

$$\left| x_p - x_1 \right| \le s_p \le S, \quad (3.3.18)$$

$$-\frac{\pi}{2} \le \varphi_1 + a\frac{i^3 S^3}{3N^3} + b\frac{i^2 S^2}{2N^2} + c\frac{iS}{N} \le \frac{\pi}{2}, \quad i = 1, \ldots, N, \quad (3.3.19)$$

де нев'язки для рівнянь (3.3.3) – (3.3.7) задаються такими функціями

$$f_1(a,b,c,S) = x_2 - x_1 - \int_0^S \cos\left(\varphi_1 + \frac{as^3}{3} + \frac{bs^2}{2} + cs\right) ds, \quad (3.3.20)$$

$$f_2(a,b,c,S) = y_2 - y_1 - \int_0^S \sin\left(\varphi_1 + \frac{as^3}{3} + \frac{bs^2}{2} + cs\right) ds\,, \qquad (3.3.21)$$

$$f_3(a,b,c,S) = \varphi_2 - \varphi_1 - \frac{aS^3}{3} - \frac{bS^2}{2} - cS\,, \qquad (3.3.22)$$

$$f_4(a,b,c,s_p) = x_p - x_1 - \int_0^{s_p} \cos\left(\varphi_1 + \frac{as^3}{3} + \frac{bs^2}{2} + cs\right) ds\,, \qquad (3.3.23)$$

$$f_5(a,b,c,s_p) = \varphi_p - \varphi_1 - \frac{as_p^3}{3} - \frac{bs_p^2}{2} - cs_p\,, \qquad (3.3.24)$$

N – кількість точок дискретизації функції $\varphi(s) = \varphi_1 + \frac{as^3}{3} + \frac{bs^2}{2} + cs$ на інтервалі $s \in [0, S]$.

Тут цільова функція (3.3.16) є негладкою та означає мінімізацію суми модулів функцій (3.3.20) – (3.3.24) – функцій нев'язок для рівнянь (3.3.3) – (3.3.7). Обмеження (3.3.17) та (3.3.18) гарантують додатні значення для довжин S та s_p, які є верхніми границями для визначених інтегралів в функціях нев'язок (3.3.20), (3.3.21) та (3.3.23). Тут $S_{\min} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$ – мінімальна відстань між точками (x_1, y_1) та (x_2, y_2), $S_{\max} > S_{\min}$ – параметри для управління нижньою та верхньою межами на S – загальну довжину кривої. Чим ближчим до $S_{\min}$ є значення верхньої межі $S_{\max}$, тим легше для системи (3.3.3) – (3.3.7) уникнути «циклічних» розв'язків (третій та четвертий розв'язки на рис. 3.3.3), а значить легше знайти розв'язок, який визначає S-подібну криву (перший розв'язок на рис. 3.3.2).

Обмеження (3.3.19) забезпечує існування єдиного глобального мінімуму для задачі (3.3.16) – (3.3.19). Воно використовує доповнення задачі (3.3.16) – (3.3.18) дискретним аналогом неперервного обмеження

$$-\frac{\pi}{2} \le \varphi(s) = \varphi_1 + \frac{as^3}{3} + \frac{bs^2}{2} + cs \le \frac{\pi}{2}, \quad s \in S. \qquad (3.3.25)$$

Обмеження (3.3.25) означає, що кути нахилу дотичних в довільній точці кривої не виходять за заданий діапазон $\varphi(s) \in \left[-\pi/2, \pi/2\right]$, тобто функція $y(x)$ на інтервалі $[x_1, x_2]$ є однозначно визначеною. Це відсікає розв'язки системи (3.3.3) – (3.3.7), для яких існують такі $\bar{x} \in [x_1, x_2]$, що ордината $y(\bar{x})$ визначається неоднозначно (друга крива із рис. 3.3.2, обидві криві із рис. 3.3.3).

Якщо в оптимізаційній задачі опустити обмеження (3.3.19), то отримаємо задачу (3.3.16) – (3.3.18), яка розглядалася в статті [2]. При великих значеннях верхньої межі $S_{\max}$ задача (3.3.16) – (3.3.18) є багатоекстремальною задачею нелінійного програмування. Кількість її глобальних мінімумів визначається значенням $S_{\max}$. Так, наприклад, якщо $S_{\max} = 2.0$, то для вихідних даних (3.3.8) задача (3.3.16) – (3.3.18) має єдину точку глобального мінімуму, дві точки глобального мінімуму – якщо $S_{\max} = 2.5$, три – якщо $S_{\max} = 2.7$, чотири – якщо $S_{\max} = 3.0$. Задача (3.3.16) – (3.3.19) має всього одну точку глобального мінімуму, яка співпадає з розв'язком системи (3.3.3) – (3.3.7) з найменшою загальною довжиною кривої.

Якщо в результаті пошуку локального мінімуму для задачі (3.3.16) – (3.3.19) отримуємо $f^* = 0$, то це означає, що знайдена точка глобального мінімуму $\left(a^*, b^*, c^*, S^*, s_p^*\right)^T$, яка є розв'язком системи (3.3.3) – (3.3.7). Якщо отримуємо $f^* > 0$, то точка $\left(a^*, b^*, c^*, S^*, s_p^*\right)^T$ не є розв'язком системи (3.3.3) – (3.3.7). Це може бути як у випадку відсутності розв'язку у системи (3.3.3) – (3.3.7) при обмеженнях (3.3.17), (3.3.18), так і у випадку, якщо алгоритм зупиниться в «неоптимальній» точці, враховуючи, що для великих значень параметра $S_{\max}$ задача (3.3.16) – (3.3.18) є багатоекстремальною.

Задача (3.3.16) – (3.3.19) є задачею мінімізації негладкої функції, яка визначена не при всіх значеннях S та s_p, а тільки при тих, які є

додатними та дозволяють обчислювати визначені інтеграли для функцій $f_1(a,b,c,S)$, $f_2(a,b,c,S)$ та $f_4(a,b,c,s_p)$. Для знаходження точки локального мінімуму у задачі (3.3.16) – (3.3.18) може бути використана модифікація *r*-алгоритму [4, 5], яка враховує вказану особливість задачі. У точці, де узагальнений градієнт цільової функції є невизначеним, модифікація r-алгоритму використовує узагальнений градієнт до одного із порушених обмежень (3.3.17), (3.3.18) і (3.3.19).

Алгоритм розв'язання задачі (3.3.16) – (3.3.19) реалізований за допомогою методу мультистарту та Octave-функції **ralgb5a** [6, 7] та знаходить найкращий локальний мінімум цільової негладкої функції за допомогою запуску модифікації *r*-алгоритму із заданої кількості стартових точок. Алгоритм використовує аналітичний спосіб обчислення узагальнених градієнтів цільової функції (3.3.16) та метод трапецій для обчислення інтегралів як у формулах (3.3.20), (3.3.21) та (3.3.23) так і у формулах для узагальнених градієнтів.

Наведемо формули для обчислення узагальненого градієнту цільової функції (3.3.16). Нехай вектор $g_f = \left(\frac{\partial f}{\partial a}, \frac{\partial f}{\partial b}, \frac{\partial f}{\partial c}, \frac{\partial f}{\partial S}, \frac{\partial f}{\partial s_p}\right)^T$ є узагальненим градієнтом цільової функції $f\left(a,b,c,S,s_p\right) = \sum_{i=1}^{3} \left| f_i\left(a,b,c,S\right)\right| + \sum_{i=4}^{5} \left| f_i\left(a,b,c,s_p\right)\right|$. Перші три компоненти узагальненого градієнта обчислюються за формулами:

$$\frac{\partial f}{\partial a} = \sum_{i=1}^{5} sign\left(f_i\right)\frac{\partial f_i}{\partial a}, \quad \frac{\partial f}{\partial b} = \sum_{i=1}^{5} sign\left(f_i\right)\frac{\partial f_i}{\partial b},$$

$$\frac{\partial f}{\partial c} = \sum_{i=1}^{5} sign\left(f_i\right)\frac{\partial f_i}{\partial c}, \qquad (3.3.26)$$

де

$$\frac{\partial f_1}{\partial a} = -\int_0^S \frac{s^3}{3}\sin\left(\varphi_1 + \frac{as^3}{3} + \frac{bs^2}{2} + cs\right)ds,$$

$$\frac{\partial f_2}{\partial a} = -\int_0^S \frac{s^3}{3}\cos\left(\varphi_1 + \frac{as^3}{3} + \frac{bs^2}{2} + cs\right)ds, \quad \frac{\partial f_3}{\partial a} = -\frac{S^3}{3}, \qquad (3.3.27)$$

$$\frac{\partial f_4}{\partial a} = -\int_0^{s_p} \frac{s^3}{3}\sin\left(\varphi_1 + \frac{as^3}{3} + \frac{bs^2}{2} + cs\right)ds, \quad \frac{\partial f_5}{\partial a} = -\frac{s_p^3}{3},$$

$$\frac{\partial f_1}{\partial b} = -\int_0^S \frac{s^2}{2}\sin\left(\varphi_1 + \frac{as^3}{3} + \frac{bs^2}{2} + cs\right)ds,$$

$$\frac{\partial f_2}{\partial b} = -\int_0^S \frac{s^2}{2}\cos\left(\varphi_1 + \frac{as^3}{3} + \frac{bs^2}{2} + cs\right)ds, \quad \frac{\partial f_3}{\partial b} = -\frac{S^2}{2}, \qquad (3.3.28)$$

$$\frac{\partial f_4}{\partial b} = -\int_0^{s_p} \frac{s^2}{2}\sin\left(\varphi_1 + \frac{as^3}{3} + \frac{bs^2}{2} + cs\right)ds, \quad \frac{\partial f_5}{\partial b} = -\frac{s_p^2}{2},$$

$$\frac{\partial f_1}{\partial c} = -\int_0^S s\sin\left(\varphi_1 + \frac{as^3}{3} + \frac{bs^2}{2} + cs\right)ds,$$

$$\frac{\partial f_2}{\partial c} = -\int_0^S s\cos\left(\varphi_1 + \frac{as^3}{3} + \frac{bs^2}{2} + cs\right)ds, \quad \frac{\partial f_3}{\partial c} = -S, \qquad (3.3.29)$$

$$\frac{\partial f_4}{\partial c} = -\int_0^{s_p} s\sin\left(\varphi_1 + \frac{as^3}{3} + \frac{bs^2}{2} + cs\right)ds, \quad \frac{\partial f_5}{\partial c} = -s_p.$$

Дві останні компоненти узагальненого градієнта обчислюються за формулами:

$$\frac{\partial f}{\partial S} = \sum_{i=1}^{3} sign(f_i)\frac{\partial f_i}{\partial S}, \quad \frac{\partial f}{\partial s_p} = \sum_{i=4}^{5} sign(f_i)\frac{\partial f_i}{\partial s_p}, \qquad (3.3.30)$$

де

$$\frac{\partial f_1}{\partial S} = -\cos\left(\varphi_1 + \frac{aS^3}{3} + \frac{bS^2}{2} + cS\right),$$

$$\frac{\partial f_2}{\partial S} = -\sin\left(\varphi_1 + \frac{aS^3}{3} + \frac{bS^2}{2} + cS\right). \qquad (3.3.31)$$

$$\frac{\partial f_3}{\partial S} = -aS^2 - bS - c, \quad \frac{\partial f_4}{\partial s_p} = \cos\left(\varphi_1 + \frac{as_p^3}{3} + \frac{bs_p^2}{2} + cs_p\right), \quad \frac{\partial f_5}{\partial s_p} = -as_p^2 - bs_p - c. \qquad (3.3.32)$$

У точці, де узагальнений градієнт цільової функції є невизначеним, для модифікації *r*-алгоритму він замінюється на узагальнений градієнт до одного із порушених двосторонніх обмежень (3.3.17), (3.3.18) або (3.3.19). Для цього використовуються шість порушених обмежень $g_i(\circ) \le 0$, $i = \overline{1,6}$, які вибираються у такому порядку: $g_1(S) = S_{\min} - S$, $g_2(S) = S - S_{\max}$ – для двосторонніх обмежень (3.3.17), $g_3(s_p) = |x_p - x_1| - s_p$, $g_4(S, s_p) = s_p - S$ – для обмежень (3.3.18), $g_5(a,b,c,S) = \max\limits_{i=1}^{N}\left\{-\frac{\pi}{2} - \varphi_1 - a\frac{i^3S^3}{3N^3} - b\frac{i^2S^2}{2N^2} - c\frac{iS}{N}\right\}$, $g_6(a,b,c,S) = \max\limits_{i=1}^{N}\left\{\varphi_1 + a\frac{i^3S^3}{3N^3} + b\frac{i^2S^2}{2N^2} + c\frac{iS}{N} - \frac{\pi}{2}\right\}$ – для двосторонніх обмежень (3.3.19). Якщо всі шість обмежень виконуються, то тоді використовується узагальнений градієнт цільової функції (3.3.16), який обчислюється за формулами (3.3.26) – (3.3.32).

Описаний алгоритм розв'язання оптимізаційної задачі (3.3.16) – (3.3.19) реалізований мовою Octave. За його допомогою можна знаходити розв'язки системи (3.3.3) – (3.3.7), яким відповідають *S*-подібні криві. Якщо, у задачі (3.3.16) – (3.3.19) опустити обмеження (3.3.19), а це рівносильно ігноруванню порушених обмежень $g_5(\circ)$ та $g_6(\circ)$, то тоді алгоритм при великих значеннях параметра $S_{\max}$ дозволяє знаходити також і «циклічні» розв'язки. Так, саме за допомогою такого алгоритму були знайдені всі чотири розв'язки, які наведені у табл. 3.3.1.

3.3.5 Три обчислювальні експерименти

У підрозділі наведено результати обчислювальних експериментів, які характеризують роботу алгоритму для тестового прикладу (3.3.8) та його масштабованого варіанту ($\mu=100$)

$$x_1 = 0,\ y_1 = 246,\ \varphi_1 = 0^\circ;\quad x_2 = 100,\ y_2 = 275,\ \varphi_2 = 0^\circ;$$
$$x_p = 70,\ \ \varphi_p = 0.209440 = 12^\circ. \tag{3.3.33}$$

Для вихідних даних (3.3.8) глобальним мінімумом задачі (3.3.16) – (3.3.19) є єдина точка, для якої компоненти $a^*, b^*, c^*, S^*, s_p^*$ співпадають з першим розв'язком системи (3.3.3) – (3.3.7) із табл. 3.3.1. Для вихідних даних (3.3.33) глобальним мінімумом є точка з компонентами $\bar{a}^* = a^* \times 10^{-6}$, $\bar{b}^* = b^* \times 10^{-4}$, $\bar{c}^* = c^* \times 10^{-2}$, $\bar{S}^* = S^* \times 100$, $\bar{s}_p^* = s_p^* \times 100$.

Перший експеримент пов'язаний з дослідженням збіжності алгоритму в залежності від вибору стартової точки та верхньої межі $S_{\max} = qS_{\min}$, де $q \in \{1.2, 1.5, 2.0, 3.0\}$. Компоненти стартової точки для коефіцієнтів квадратичної функції кривини a_0, b_0, c_0 генерувались за допомогою Octave-датчика випадкових чисел rand("seed", 2020), а для довжин S та s_p вибирались рівними їх нижнім межам $S_0 = S_{\min} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$, $(s_p)_0 = |x_p - x_1|$. Критерієм успішного розв'язання задачі (3.3.16) – (3.3.19) була вибрана умова, що значення функції, яка мінімізується, є меншим ніж 10^{-6}. Найбільш значимі параметри *r*-алгоритму вибиралися відповідно до рекомендованих в [8] для мінімізації негладких функцій – коефіцієнт розтягу простору $\alpha = 1.5$, величина початкового кроку $h_0 = 1.0$, коефіцієнт зменшення кроку $q_0 = 1.0$, точність зупинки по аргументу $\varepsilon_x = 10^{-6}$, максимальна кількість ітерацій maxitn=500.

У табл. 3.3.2 наведено результати першого експерименту щодо розв'язання задачі (3.3.16) – (3.3.19) для 100 стартових точок, де компоненти a_0, b_0, c_0 належали чотирьом діапазонам $[-5,5]^3$, $[-25,25]^3$, $[-50,50]^3$, $[-100,100]^3$. Розглядалися три різних сценарії. Перший сценарій – задача вирішувалась для вихідних даних (3.3.8), другий та третій сценарії – задача вирішувалась задача для вихідних даних (3.3.33). В першому та другому сценаріях стартові точки вибирались одними і тими ж, а для третього сценарію перші три компоненти стартових точок домножувалися на коефіцієнти $\left\{10^{-6},10^{-4},10^{-2}\right\}$, які узгоджуються з коефіцієнтом масштабу $\mu = 100$ для вихідних даних (3.2.33).

Табл. 3.3.2. Кількість успішних стартових точок для трьох сценаріїв (перший експеримент, $\varphi_1 = \varphi_2 = 0^\circ$, $\varphi_p = 0.209440 = 12^\circ$)

діапазон	$q = 1.2$	$q = 1.5$	$q = 2.0$	$q = 3.0$
$x_1 = 0,\ y_1 = 2.46,\quad x_2 = 1,\ y_2 = 2.75,\quad x_p = 0.7$				
$[-5,5]^3$	100/100	100/100	100/100	99/100
$[-25,25]^3$	100/100	100/100	88/100	80/100
$[-50,50]^3$	93/100	84/100	74/100	71/100
$[-100,100]^3$	81/100	70/100	65/100	67/100
$x_1 = 0,\ y_1 = 246,\quad x_2 = 100,\ y_2 = 275,\quad x_p = 70$				
$[-5,5]^3$	95/100	95/100	95/100	95/100
$[-25,25]^3$	87/100	88/100	85/100	81/100
$[-50,50]^3$	86/100	70/100	70/100	73/100
$[-100,100]^3$	78/100	64/100	60/100	62/100

діапазон	$q=1.2$	$q=1.5$	$q=2.0$	$q=3.0$
$x_1=0,\ y_1=2.46,\quad x_2=1,\ y_2=2.75,\quad x_p=0.7$				
$x_1=0,\ y_1=246,\ x_2=100,\ y_2=275,\ x_p=70,\ D=\mathrm{diag}\left\{10^{-6},10^{-4},10^{-2}\right\}$				
$[-5,5]^3\times D$	98/100	98/100	98/100	98/100
$[-25,25]^3\times D$	95/100	95/100	95/100	95/100
$[-50,50]^3\times D$	92/100	92/100	92/100	92/100
$[-100,100]^3\times D$	95/100	95/100	95/100	95/100

Із табл. 3.3.2 видно, що для добре масштабованих даних із 100 запусків алгоритм завжди збігається до точки глобального мінімуму задачі (3.3.16) – (3.3.19), якщо перші три компоненти стартової точки належать діапазонам $[-5,5]^3$ та $[-25,25]^3$. Якщо перші три компоненти належать діапазонам $[-50,50]^3$ та $[-100,100]^3$, то алгоритм не завжди збігається до точки глобального мінімуму. Наприклад, для $S_{\max}=2\times S_{\min}$ та діапазону $[-100,100]^3$ алгоритм знайшов розв'язок в шістдесяти п'яти випадках із ста, а в тридцяти п'яти випадках алгоритм не зміг знайти розв'язку та зупинився в точках, де значення цільової функції (3.3.16) є більшим нуля. Це зумовлено тим, що глобальний мінімум у задачі (3.3.16) – (3.3.19) єдиний і йому відповідає довжина $S^*=1.05562$, а стартова точка далека від точки мінімуму, завдяки чому градієнтний процес прямує до іншого розв'язку системи, якому відповідає суттєво більша довжина кривої. Такі розв'язки можна знайти, якщо вибрати значно завищеною верхню межу $S_{\max}$ та розв'язувати задачу (3.3.16) – (3.3.18), тобто не враховувати обмеження (3.3.19). Але цим розв'язкам відповідають «циклічні» криві (рис. 3.3.3), які не підходять для моделювання зовнішнього контуру сопла Франкля.

Для другого та третього сценаріїв, коли вихідні дані (3.3.33) погано масштабовані, немає ні одного випадку, щоб алгоритму

вдалося успішно розв'язати задачу (3.3.16) – (3.3.18) у всіх випадках із ста. При цьому результати для третього сценарію є кращими за результати для другого, що пояснюється тим, що для третього сценарію стартова точка є ближчою до точки глобального мінімуму, ніж для другого сценарію. З результатів першого експерименту випливає, що краще розв'язувати задачу з добре масштабованими даними, ніж з погано масштабованими даними, а стартову точку вибирати такою, щоб її перших три компоненти були якомога ближчими до нуля. Це підтверджують результати другого та третього експериментів, які наведені у табл. 3.3.3 та 3.3.4.

Табл. 3.3.3. Прискорення *r*-алгоритму (другий експеримент)

ε_x	$q=1.2$	$q=1.5$	$q=2.0$	$q=3.0$
$\alpha=1.5, \quad h_0=1.0, \quad q_1=1.0$				
10^{-4}	133/152/20	142/162/17	142/162/17	142/162/17
10^{-6}	198/243/20	197/232/17	197/232/17	197/232/17
10^{-8}	256/307/20	253/312/17	253/312/17	253/312/17
10^{-10}	302/366/20	303/372/17	314/385/17	289/355/17
10^{-12}	371/457/20	372/465/17	365/450/17	370/457/17
$\alpha=2.0, \quad h_0=1.0, \quad q_1=0.95$				
10^{-4}	79/130/19	70/116/11	70/116/11	70/116/11
10^{-6}	113/179/19	103/154/11	103/154/11	103/154/11
10^{-8}	133/208/19	127/182/11	127/182/11	127/182/11
10^{-10}	165/258/19	146/205/11	146/205/11	146/205/11
10^{-12}	190/292/19	180/250/11	178/247/11	178/247/11

У табл. 3.3.3 наведені витрати *r*-алгоритму (кількість ітерацій / кількість обчислень узагальненого градієнта цільової функції / кількість обчислень узагальненого градієнта до порушених обмежень)

в залежності від вибору параметрів α, h_0, q_1 для знаходження *S*-подібної кривої для вихідних даних (3.8) з 5-ма різними критеріями зупинки за аргументом $\varepsilon_x = \left\{10^{-4}, 10^{-6}, 10^{-8}, 10^{-10}, 10^{-12}\right\}$. Алгоритм в обох випадках стартував з однієї і тієї ж точки з компонентами $a_0 = b_0 = c_0 = 0$, $S_0 = \sqrt{\left(x_2 - x_1\right)^2 + \left(y_2 - y_1\right)^2}$, $\left(s_p\right)_0 = \left|x_p - x_1\right|$. З таблиці видно, що завдяки вибору параметрів $\alpha = 2.0$, $h_0 = 1.0$, $q_1 = 0.95$ можна досягнути прискорення майже в два рази по відношенню до вибору параметрів $\alpha = 1.5$, $h_0 = 1.0$, $q_1 = 1.0$. Зауважимо, що хоча $q_1 = 0.95$ не рекомендується вибирати для мінімізації негладких функцій, в даному випадку його можна використовувати, оскільки кількість невідомих є невеликою.

У табл. 3.3.4 порівнюються витрати *r*-алгоритму (за кількістю ітерацій – *itn*, за кількістю обчислень узагальненого градієнта цільової функції – *nfg*, за кількістю обчислень узагальненого градієнта до порушених обмежень – *ng*) для знаходження *S*-подібної кривої для добре масштабованих даних (3.3.8) та погано масштабованих даних (3.3.33). Витрати (для (3.3.8) / для (3.3.33)) наведені для п'ятьох критеріїв зупинки за аргументом $\varepsilon_x = \left\{10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}, 10^{-9}\right\}$. В табл. 3.3.4 також наведені абсолютні відхилення між компонентами розв'язків для добре масштабованих та погано масштабованих даних. Вони обчислювались за формулами $\Delta a = \left|a^* - \bar{a}^* \times 10^6\right| \Big/ \left|a^*\right|$, $\Delta b = \left|b^* - \bar{b}^* \times 10^6\right| \Big/ \left|b^*\right|$, $\Delta c = \left|c^* - \bar{c}^* \times 10^6\right| \Big/ \left|c^*\right|$, $\Delta S = \left|S^* - \bar{S}^* \times 10^6\right| \Big/ \left|S^*\right|$, $\Delta s_p = \left|s_p^* - \bar{s}_p^* \times 10^6\right| \Big/ \left|s_p^*\right|$. З табл. 3.3.4 видно, що при розв'язанні задачі з добре масштабованими даними потрібно майже в два рази менше ітерацій та майже в чотири рази менше обчислень узагальненого градієнта, ніж при розв'язанні задачі з погано масштабованими даними.

Табл. 3.3.4. Витрати *r*-алгоритму на пошук розв'язків добре та погано масштабованих задач (третій експеримент)

	$\varepsilon_x = 10^{-5}$	$\varepsilon_x = 10^{-6}$	$\varepsilon_x = 10^{-7}$	$\varepsilon_x = 10^{-8}$	$\varepsilon_x = 10^{-9}$
itn	84/204	113/219	113/238	133/247	137/258
nfg	139/635	179/653	179/684	208/695	212/718
ng	19/51	19/51	19/51	19/51	19/51
Δa	2.64e-06	7.07e-07	2.87e-08	3.00e-09	1.06e-09
Δb	7.82e-07	5.73e-07	2.19e-08	2.09e-09	8.54e-10
Δc	1.67e-07	4.10e-07	1.58e-08	1.43e-09	6.10e-10
ΔS	2.26e-08	1.07e-08	5.70e-09	1.37e-10	6.23e-11
Δs_p	1.84e-07	2.25e-08	3.34e-09	6.72e-11	1.40e-10

Результати обчислювальних експериментів показують, що стартову точку можна вибирати рівною $\left(0,0,0,S_{\min},\left|x_p - x_1\right|\right)^T$ незалежно від масштабу даних задачі (3.3.16) – (3.3.19). Щоб знайти розв'язок із заданою точністю за менший час краще розв'язувати задачу з добре масштабованими даними, ніж з погано масштабованими даними. Масштабування може відігравати суттєву роль для вибору того чи іншого параметру при моделюванні профілів в ітераційному режимі, враховуючи, що масштаб достатньо встановити всього один раз, а задачу потрібно вирішувати багато разів.

3.3.6 Геометричне моделювання зовнішнього контуру сопла Франкля

Наведемо результати обчислювальних експериментів щодо використання розробленого алгоритму для побудови фрагментів дозвукової та надзвукової частин зовнішнього контуру сопла Франкля.

Вони пов'язані з проектуванням сопла з центральним тілом для трьох заданих реперних точок зовнішнього контуру, які визначають дозвукову та надзвукову частини сопла та їх стикування у точці критичного перерізу. В реперних точках похідні (кути нахилу дотичних) дорівнюють нулю. Для даного контуру задачі (3.3.16) – (3.3.19) визначаються вихідними даними, для яких виконується умова $|y_2 - y_1|/|x_2 - x_1| \ll 1$, тому відповідні фрагментам контурів *S*-подібні криві є витягнутими відносно горизонтальної осі.

Перший експеримент пов'язаний з дослідженням впливу кута φ_p на форму *S*-подібної кривої для тестового прикладу (3.3.8), для якого $|y_2 - y_1|/|x_2 - x_1| = 0,29$. В таблиці 3.3.5 представлені точки глобальних мінімумів трьох задач для різних значень кутів $\varphi_p \in \{16^\circ, 12^\circ, 8^\circ\}$. Використовуючи ці точки за формулами (3.3.10) легко знайти розв'язки системи (3.3.3) – (3.3.7) для масштабованих вихідних даних (3.3.33), де $\mu = 100$.

Табл. 3.3.5. Глобальні мінімуми трьох задач для вихідних даних (3.3.8)

	$\varphi_p = 16^\circ$	$\varphi_p = 12^\circ$	$\varphi_p = 8^\circ$
a^*	4.0903	7.9282	11.5304
b^*	-7.3628	-11.4065	-15.2523
c^*	2.3632	3.0756	3.7636
S^*	1.0511	1.0556	1.0630
s_p^*	0.74714	0.75399	0.76259

Розв'язки з табл. 3.3.5 проілюстровано на рис. 3.3.4, де ліворуч наведено графіки кривих, а праворуч – графіки відповідних кривин. З графіків кривин видно, що для першої та другої двохланкових кривих знаки кривини змінюються тільки в одній точці (точка перегину), що

доводить їх *S*-подібність. Для третьої кривої знак кривини змінюється в двох точках (відмічені на рис. праворуч), тому вона не є *S*-подібною. Зауважимо, що із графіків кривих такий висновок зробити неможливо, оскільки усі три криві виглядають як *S*-подібні.

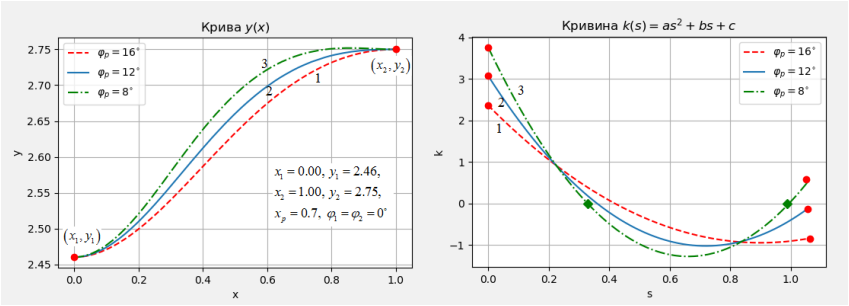


Рис. 3.3.4. Графіки кривих та кривин для трьох розв'язків з табл. 3.3.5

Другий експеримент пов'язаний з побудовою двох варіантів кривих для дозвукової та надзвукової частин зовнішнього контуру сопла за трьома заданими реперними точками: в точці $A\left(x_A = 0,\ y_A = 295\right)$ починається дозвукова частина, точка $B\left(x_B = 216,\ y_B = 246\right)$ є точкою критичного перерізу (число Маха $M = 1$), в точці $C\left(x_C = 638,\ y_C = 275\right)$ закінчується надзвукова частина. В табл. 3.3.6 наведені точки глобальних мінімумів для задач (3.3.16) – (3.3.19) з вихідними даними, де координати точок зменшувалися в $\mu = 100$ раз. Розглядалися два варіанти кутів φ_p при абсцисах x_p, яка для дозвукової частини вибиралась ближчою до абсциси точки A, а для надзвукової частини – ближчою до абсциси точки C.

Табл. 3.3.6. Розв'язки для дозвукової $\left(x_1 = 0.00,\ y_1 = 2.95,\ x_2 = 2.16,\ y_2 = 2.46,\ x_p = 0.7\right)$ та надзвукової $\left(x_1 = 2.16,\ y_1 = 2.46,\ x_2 = 6.38,\ y_2 = 2.75,\ x_p = 5.0\right)$ частин сопла

	Дозвукова частина		Надзвукова частина	
	$\varphi_p = -12^\circ$	$\varphi_p = -8^\circ$	$\varphi_p = 4^\circ$	$\varphi_p = 2^\circ$
a^*	0.53981	0.97384	0.021423	0.057183
b^*	-0.66700	-1.65380	-0.13663	-0.28816
c^*	-0.15169	0.22102	0.16123	0.26827
S^*	2.2312	2.2439	4.2327	4.2372
s_p^*	0.70440	0.70109	2.8517	2.8571

Графічна ілюстрація розв’язків наведена на рис. 3.3.5, де у верхній частині рисунку представлено графіки кривих для дозвукового та надзвукового фрагментів контуру, а в нижній частині – графіки кривин, які відповідають цим кривим. Перші криві для обох частин контуру є *S*-подібними, для них кривини змінюють знак тільки в одній точці. Другі криві не є *S*-подібними, для них кривина змінює знак в двох точках (відмічені на рис. 3.3.5 внизу).

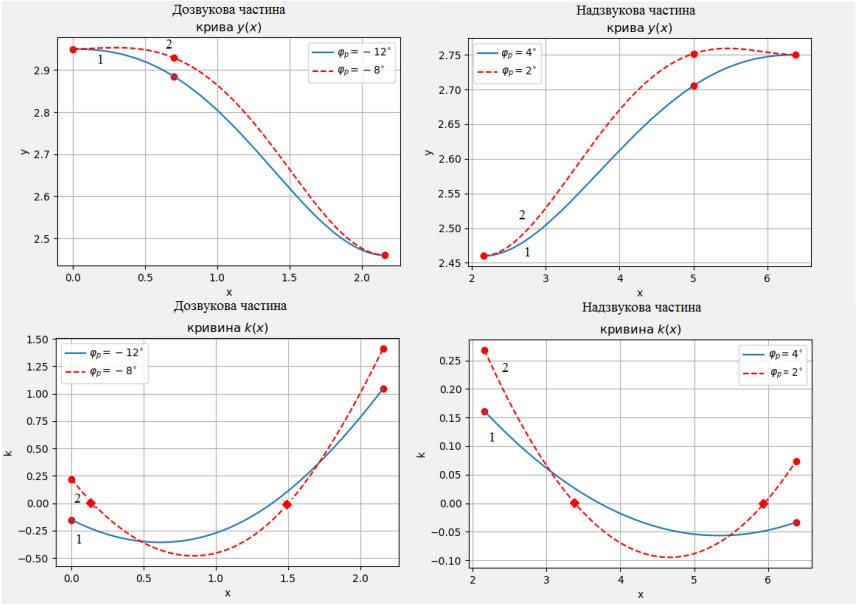


Рис. 3.3.5. Графіки кривих та кривин для розв’язків з таблиці 3.3.6

Проведені обчислювальні експерименти для проектування фрагментів дозвукової та надзвукової частини зовнішнього контуру сопла Франкля показали ефективність розробленого алгоритму для побудови двохланкових S-подібних кривих.

3.3.7 Опис програмного забезпечення

Загальна структура комплексу програм для проектування фрагментів дозвукової та надзвукової частини зовнішнього контуру сопла Франкля має вигляд, як на рис. 3.3.6. Комплекс програм включає головну програму **Frankl-quadratic**, яка за допомогою *r*-алгоритму розв'язує задачу (3.3.16) – (3.3.19), та три підпрограми: **ralgb5a** (реалізує *r*-алгоритм), **fgO1** (обчислює значення функції та узагальненого градієнта) та **plotO1** (забезпечує графічний вигляд знайденої кривої та її характеристик). Нижче наведемо їх короткий

опис та коди мовою Octave, за винятком підпрограми ralgb5a. Її опис та Octave код, що включає і короткі англомовні коментарі для вхідних та вихідних параметрів, наведено в [9, ст. 26–28].

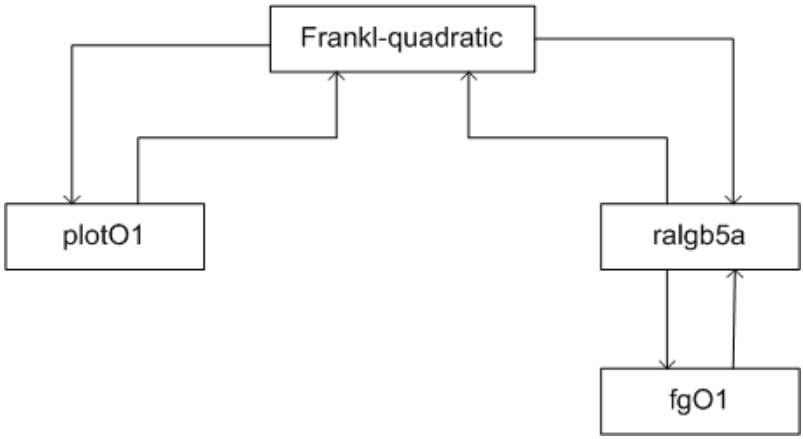


Рис. 3.3.6. Блок-схема програми Frankl-quadratic

Головна програма Frankl-quadratic керує процесом знаходження розв'язку оптимізаційної задачі (3.3.16) – (3.3.19). Її код є наступним:

```
global x1 y1 phi1 x2 y2 phi2 xp phip
global N Smin Smax spmin
global nfgr

# test  No 1
#x1 = 0.0;  y1 = 2.46;  phi1 = 0.0;
#x2 = 1.0;  y2 = 2.75;  phi2 = 0.0;
#xp = 0.7;  phip = (9.0/180.0)*pi;

# test  No 1m
#x1 =   0.0;  y1 = 246.0;  phi1 = 0.0;
#x2 = 100.0;  y2 = 275.0;  phi2 = 0.0;
#xp =  70.0;  phip = (9.0/180.0)*pi;

# test  No 2
x1 = 0;   y1 = 2.75; phi1 = 0.0;
```

```
x2 = 1.;  y2 = 2.46; phi2 = 0.0;
xp = 0.3; phip = -(12.0/180.0)*pi;

# test  No 2m
#x1 =   0.0;  y1 = 275.0;  phi1 = 0.0;
#x2 = 100.0;  y2 = 246.0;  phi2 = 0.0;
#xp =  30.0;  phip = -(12.0/180.0)*pi;

printf("x1 y1 phi1 %13.5e %13.5e %13.5e = %13.5e\n",
        x1, y1, phi1, 180*(phi1/pi));
printf("x2 y2 phi2 %13.5e %13.5e %13.5e = %13.5e\n",
        x2, y2, phi2, 180*(phi2/pi));
printf("xp phip %13.5e %13.5e = %13.5e\n",
        xp, phip, 180*(phip/pi));

Smin = sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2));
spmin = xp - x1; Smax = 1.2*Smin;

printf("\n");
alpha = 2.0, h0 = 1.0, q1 = 0.95,
epsx = 1.e-6, epsg = 1.e-8, maxitn = 1000, intp=20;

printf("\n");
N = 200;  eps=1.d-4;

nfgr = 0; x0 = zeros(3,1);
x0(4,1) = Smin; x0(5,1) = spmin;
printf("x0(tr) = %9.5f %9.5f %9.5f %9.5f %9.5f \n",
        x0(1:5));
[xr,fr,itn,nfg,istop] = ralgb5a(@fgO1,x0,alpha,h0,q1,
                        epsg,epsx,maxitn,intp);
printf("..itn %4d  fr %23.15e istop %d  nfg(-) %4d(%3d)\n",
        itn, fr, istop, nfg, nfg-nfgr);
if (fr >= eps)
  printf(" error: fr = %9.2e > eps = %9.2e \n",fr,eps);
endif

a=xr(1,1); b=xr(2,1); c=xr(3,1); S=xr(4,1); sp=xr(5,1);
printf("\n");
printf(" fr %9.2e\n",fr);
printf("a b c S sp  %13.5e %13.5e %13.5e %13.5e %13.5e\n",
        a,b,c,S,sp);
[dphi2] = pict1(a,b,c,S,sp,200);
print -deps figure-fgO1a.eps
print -dpng figure-fgO1a.png
pause
```

В програмі задаються вхідні дані, встановлюються параметри *r*-алгоритму, запускається процесс оптимізації для розв'язання задачі (3.3.16) – (3.3.19), аналізується отриманий розв'язок (перевіряється чи нев'язка більша заданої величини). Компоненти стартової точки для коефіцієнтів квадратичної функції вибираються рівними нулю. Результати розрахунку виводяться у файл протоколу роботи програми та відображаються у вигляді графічного вікна, яке включає графіки кривих та графіки відповідних кривин. Приклади графічного вікна наведено на рис. 3.3.7 та рис. 3.3.8.

Програма Frankl-quadratic використовує підпрограми ralgb5a та plotO1. Підпрограма ralgb5a реалізує роботу *r*-алгоритму для розв'язання оптимізаційної задачі (3.3.16) – (3.3.19), використовуючи підпрограму fgO1, код якої наведено нижче.

```
function [f,g] = fgO1(x)
global x1 y1 phi1 x2 y2 phi2 xp phip
global N Smin Smax spmin
global nfgr

a3=x(1,1)/3;  b2=x(2,1)/2;  c=x(3,1);  S=x(4,1); sp=x(5,1);

f = inf; g = zeros(rows(x),1);
if((Smin-S)>0) g(4,1)=-1; return; endif
if((S-Smax)>0) g(4,1)=1; return; endif

if((spmin-sp)>0) g(5,1)=-1; return; endif
if((sp-S)>0) g(4,1)=-1; g(5,1)=1; return; endif

ds = S/N; ss = ds*[1:N];
phi = phi1*ones(1,N) + ((a3*ss+b2).*ss+c).*ss;
[phiA imin] = min(phi); sA=ss(imin);
if((-pi/2-phiA)>0)
   g(1,1)=-sA*sA*sA/3; g(2,1)=-sA*sA/2; g(3,1)=-sA; return;
endif
if((phiA-pi/2)>0)
   g(1,1)=sA*sA*sA/3; g(2,1)=sA*sA/2; g(3,1)=sA; return;
endif
[phiB imax] = max(phi); sB=ss(imax);
if((-pi/2-phiB)>0)
   g(1,1)=-sB*sB*sB/3; g(2,1)=-sB*sB/2; g(3,1)=-sB; return;
endif
if((phiB-pi/2)>0)
```

```
   g(1,1)=sB*sB*sB/3; g(2,1)=sB*sB/2; g(3,1)=sB; return;
endif

# We calculate f = abs(x2-tmpx-x1) + abs(y2-tmpy-y1);

fcos = cos(phi); fsin = sin(phi);
tmpx = sum(fcos) + cos(phi1)/2 - fcos(1,N)/2;
tmpx = ds*tmpx;
tmpy = sum(fsin) + sin(phi1)/2 - fsin(1,N)/2;
tmpy = ds*tmpy;
tmp1 = x2 - tmpx - x1; tmp2 = y2 - tmpy - y1;
f = abs(tmp1) + abs(tmp2);
g(4,1) = - sign(tmp1)*fcos(1,N) - sign(tmp2)*fsin(1,N);

tmpsin = fsin.*ss;  tmpcos = fcos.*ss;
tmpc1 = sum(tmpsin) - tmpsin(1,N)/2;  tmpc1 = ds*tmpc1;
tmpc2 = sum(tmpcos) - tmpcos(1,N)/2;  tmpc2 = ds*tmpc2;
g(3,1) = sign(tmp1)*tmpc1 - sign(tmp2)*tmpc2;

tmpsin = tmpsin.*ss;  tmpcos = tmpcos.*ss;
tmpb1 = sum(tmpsin) - tmpsin(1,N)/2;  tmpb1=ds*tmpb1/2;
tmpb2 = sum(tmpcos) - tmpcos(1,N)/2;  tmpb2=ds*tmpb2/2;
g(2,1) = sign(tmp1)*tmpb1 - sign(tmp2)*tmpb2;

tmpsin = tmpsin.*ss;  tmpcos = tmpcos.*ss;
tmpa1 = sum(tmpsin) - tmpsin(1,N)/2;  tmpa1=ds*tmpa1/3;
tmpa2 = sum(tmpcos) - tmpcos(1,N)/2;  tmpa2=ds*tmpa2/3;
g(1,1) = sign(tmp1)*tmpa1 - sign(tmp2)*tmpa2;

# We calculate f = f + abs(phi2 - phi(S))
tmp3 = phi2 - ((a3*S+b2)*S+c)*S - phi1; f = f + abs(tmp3);
g(1,1) = g(1,1) - sign(tmp3)*S*S*S/3;
g(2,1) = g(2,1) - sign(tmp3)*S*S/2;
g(3,1) = g(3,1) - sign(tmp3)*S;
g(4,1) = g(4,1) - sign(tmp3)*(x(1,1)*S*S +
         x(2,1)*S + x(3,1));

# We calculate f = f + abs(xp-tmpx-x1);
ds = sp/N; ss = ds*[1:N];
phi = phi1*ones(1,N) + ((a3*ss+b2).*ss+c).*ss;

fcos = cos(phi); fsin = sin(phi);
tmpx = sum(fcos) + cos(phi1)/2 - fcos(1,N)/2;
tmpx = ds*tmpx;
tmp4 = xp - tmpx - x1; f = f + abs(tmp4);
g(5,1) = g(5,1) - sign(tmp4)*fcos(1,N);
```

```
tmpsin = fsin.*ss;
tmpc1 = sum(tmpsin) - tmpsin(1,N)/2;  tmpc1 = ds*tmpc1;
g(3,1) = g(3,1) + sign(tmp4)*tmpc1;

tmpsin = tmpsin.*ss;
tmpb1 = sum(tmpsin) - tmpsin(1,N)/2;  tmpb1=ds*tmpb1/2;
g(2,1) = g(2,1) + sign(tmp4)*tmpb1;

tmpsin = tmpsin.*ss;
tmpa1 = sum(tmpsin) - tmpsin(1,N)/2;  tmpa1=ds*tmpa1/3;
g(1,1) = g(1,1) + sign(tmp4)*tmpa1;

# We calculate f = f + abs(phip - phi(sp))
tmp5 = phip - ((a3*sp+b2)*sp+c)*sp - phi1;
f = f + abs(tmp5);
g(1,1) = g(1,1) - sign(tmp5)*sp*sp*sp/3;
g(2,1) = g(2,1) - sign(tmp5)*sp*sp/2;
g(3,1) = g(3,1) - sign(tmp5)*sp;
g(5,1) = g(5,1) -
         sign(tmp5)*(x(1,1)*sp*sp+x(2,1)*sp+x(3,1));

nfgr = nfgr + 1;
endfunction
```

Підпрограма plotO1 відображає вхідні дані задачі в графічне вікно. Її код є наступним:

```
function [dphi2] = pict1(a,b,c,S,s1,N)

global x1 y1 phi1 x2 y2 phi2 xp phip

s=0; ds=S/N;
x(1,1)=x1;  y(1,1)=y1;  ss(1)=s;  k(1)=c;  dk(1)=b;  phi(1)  =
phi1;
tempx  =  0;  tempy  =  0;  t1  =  phi1;  fx1  =  cos(t1);  fy1  =
sin(t1);
for i=1:N
   s=s+ds; t1 = phi1 + a*s*s*s/3 + b*s*s/2 + c*s;
   fx2 = cos(t1); fy2 = sin(t1);
   tempx = tempx + ds*(fx1+fx2)/2.0; fx1=fx2;
   tempy = tempy + ds*(fy1+fy2)/2.0; fy1=fy2;
   x(i+1,1)= x1 + tempx; y(i+1,1) = y1 + tempy;
   ss(i+1) = s; k(i+1) = a*s*s + b*s + c; dk(i+1) = 2*a*s +
b;
   phi(i+1) = phi1 + a*s*s*s/3 + b*s*s/2 + c*s;
```

```
endfor
dphi2 = phi(N+1) - phi2;

s=0; ds=s1/N;
xx(1,1)=x1;yy(1,1)=y1;                sss(1)=s;phi_1(1)=phi1;
kk(1)=c;dkk(1)=b; tempx = 0; tempy = 0; t1 = phi1; fx1 =
cos(t1); fy1 = sin(t1);
for i=1:N
   s=s+ds; t1 = phi1 + a*s*s*s/3 + b*s*s/2 + c*s;
   fx2 = cos(t1); fy2 = sin(t1);
   tempx = tempx + ds*(fx1+fx2)/2.0; fx1=fx2;
   tempy = tempy + ds*(fy1+fy2)/2.0; fy1=fy2;
   xx(i+1,1)= x1 + tempx; yy(i+1,1) = y1 + tempy;
   sss(i+1) = s; kk(i+1) = a*s*s + b*s + c;
   dkk(i+1) = 2*a*s + b;
   phi_1(i+1) = phi1 + a*s*s*s/3 + b*s*s/2 + c*s;
endfor

hfig1=figure; figure(hfig1); #hold on;
subplot ( 2 , 2 , 1);
plot(x(1:(N+1),1),y(1:(N+1),1),'b','linewidth',2,
     x1,y1,'ro',xx(N+1,1),yy(N+1,1),'ro',x2,y2,'or');
title('y(x)'); grid('on');
axis equal;
subplot ( 2 , 2 , 2 );
plot(ss(1:(N+1)),phi(1:(N+1)),'r','linewidth',2,
     ss(1),phi(1),'b',sss(N+1),phi_1(N+1),'bo',
     ss(N+1),phi(N+1),'bo');
title('\phi(s)'); grid('on');
subplot ( 2 , 2 , 3 );
plot(ss(1:(N+1)),k(1:(N+1)),'g','linewidth',2,
     ss(1),k(1),'ro',ss(N+1),kk(N+1),'ro',
     ss(N+1),k(N+1),'ro');
title('k(s)'); grid('on');
subplot ( 2 , 2 , 4 );
plot(ss(1:(N+1)),dk(1:(N+1)),'g','linewidth',2,
     ss(1),dk(1),'ro',sss(N+1),dkk(N+1),'ro',
     ss(N+1),dk(N+1),'ro');
title('k''(s)'); grid('on');

endfunction
```

За допомогою програми Frankl-quadratic був розраховані фрагменти зовнішнього контуру сопла (рис. 3.3.7 та рис. 3.3.8), який вибраний за стартовий для побудови контуру центрального тіла.

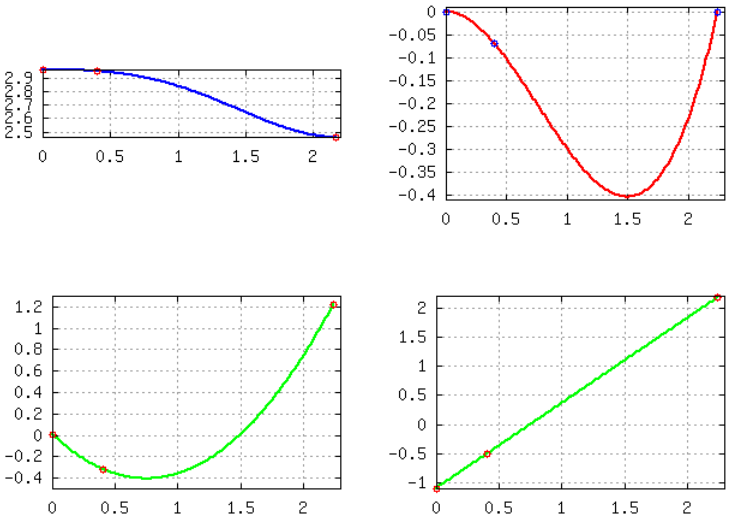


Рис. 3.3.7. Звужуюча частина зовнішнього контуру: $\mu = 100$

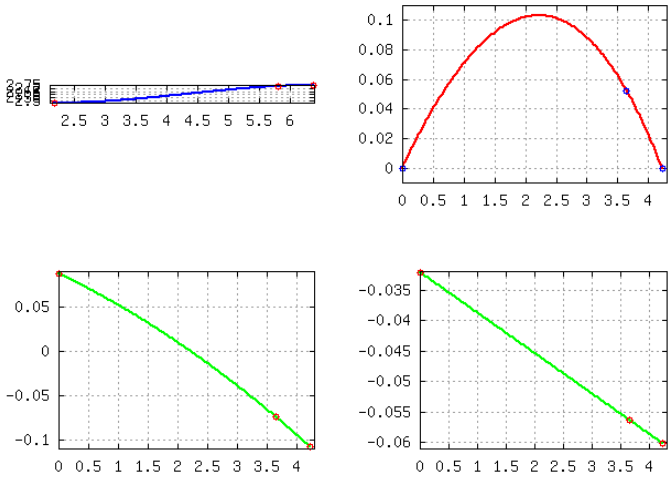


Рис. 3.3.8. Розширююча частина зовнішнього контуру: $\mu = 100$

Для дозвукової частини зовнішнього контуру використовувались наступні вхідні параметри: $x_1 = 0.0,\ y_1 = 296,\ \ x_2 = 216.68,\ y_2 = 246.041,$ $x_p = 40.0,\ \varphi_1 = -4^\circ$, а для побудови надзвукової частини сопла використовувались такі параметри: $x_1 = 216.68,\ y_1 = 246.041,$ $x_2 = 638.0,\ y_2 = 275.0,,\ x_p = 580.0,\ \varphi_2 = 3^\circ$.

3.3.8 Висновки

Розроблено математичну модель, алгоритм та програмне забезпечення для задачі побудови *S*-подібної кривої в натуральній параметризації, яка проходить через дві задані точки із заданими кутами нахилу дотичних у них та забезпечує заданий кут нахилу дотичної в точці із заданою абсцисою. Наведена система нелінійних інтегральних рівнянь для квадратичної кривини, досліджено її властивості, описано відповідну задачу мінімізації негладкої функції та алгоритм її розв'язання. Алгоритм основано на модифікації методу з розтягом простору в напрямі різниці двох послідовних узагальнених градієнтів. Проведено обчислювальні експерименти, які показали ефективність розробленого алгоритму для проектування фрагментів дозвукової та надзвукової частин зовнішнього контуру сопла Франкля.

Розроблений алгоритм побудови зовнішнього контуру сопла за допомогою *S* -подібних кривих з використанням натуральної параметризації дозволяє управляти формою контуру за допомогою мінімальної кількості параметрів (кути нахилу дотичних в точках з відомими абсцисами у дозвуковій та надзвуковій частинах). При цьому кривина контуру змінюється плавно, забезпечуючи тим самим, зокрема, необхідні геометричні та газодинамічні властивості контуру. Обчислювальні експерименти підтверджують стійку роботу алгоритму для добре масштабованих вихідних даних контуру, який проектується. Використовуючи отриманий розв’язок за формулами (3.3.10) легко знайти розв’язки з необхідною точністю для погано масштабованих вихідних даних.

Розроблені математична модель, алгоритм та програмне забезпечення можуть бути використані для профілювання фрагментів сопел та перехідних каналів реактивних двигунів [8]. Поряд з методом Безьє – Бернштейна розроблений алгоритм можна використовувати для побудови фрагментів профілів перехідних каналів змінного перерізу з необхідними геометричними властивостями. Так, наприклад, за його допомогою можна моделювати фрагменти профілів, які представляються опуклими (увігнутими) функціями, як монотонно зростаючими, так і монотонно спадними. Вибір абсциси x_p та кута φ_p дозволяє управляти формою кривої таким чином, щоб в базисних (реперних) точках характеристики кривої відповідали заданим характеристикам профілю, що проектується.

Розроблені алгоритм та програмне забезпечення використовується для побудови зовнішнього контуру сопла з центральним тілом [10].

Список літератури

1. Стецюк П.І., Ткаченко О.В., Хом'як О.М., Грицай О.Л. Побудова зовнішнього контуру сопла Франкля з використанням S-подібних кривих із квадратичним законом розподілу кривини. Кибернетика и системный анализ. 2020. № 6. С. 120–135.
2. Стецюк П.І., Ткаченко О.В., Грицай О.Л. До побудови зовнішнього контура сопла Франкля за квадратичною кривиною. Кібернетика та комп'ютерні технології. 2020. № 1. С. 23–31.
3. Борисенко В.Д., Устенко С.А., Устенко І.В. Геометрическое моделирование s-образных скелетных линий профилей лопаток осевых компрессоров. Вестник двигателестроения. 2018. №1. С. 45–52. https://doi.org/10.15588/1727-0219-2018-1-7
4. Шор Н.З., Стецюк П.И. Использование модификации r-алгоритма для нахождения глобального минимума

полиномиальных функций. Кибернетика и систем. анализ. 1997. № 4. С. 28–49.
5. Stetsyuk P.I. Shor’s r-Algorithms: Theory and Practice. In: Optimization Methods and Applications: In Honor of the 80th Birthday of Ivan V. Sergienko. Ed. by Butenko S., Pardalos P.M, Shylo V. Springer. 2017. Pp. 495–520.
6. Стецюк П.И. Теория и программные реализации r-алгоритмов Шора. Кибернетика и системный анализ. 2017. № 5. С. 43–57.
7. Стецюк П.І. Комп'ютерна програма "Octave-програма ralgb5a: r(α)-алгоритм з адаптивним кроком". Свідоцтво про реєстрацію авторського права на твір № 85010. Україна. Міністерство економічного розвитку і торгівлі. Державний департамент інтелектуальної власності. Дата реєстрації 29.01.2019.
8. Крайко А.А. Профилирование сопел и переходных каналов реактивных двигателей: дис. кандидата физико-математических наук: 01.02.05 / Крайко Алла Александровна. М., 2014. 151 с.
9. Розроблення комплексу програм побудови теоретичних контурів зовнішньої і внутрішньої поверхонь сопла з центральним тілом по заданому закону зміни площ. І.В. Сергієнко, П.І. Стецюк, О.М. Литвин та інші. Заключний звіт про науково-дослідну роботу № держ. реєстрації 0118U006687. К.: Ін-т кібернетики імені В.М. Глушкова НАН України, 2018. 51 с.
10. Розроблення комплексу програм побудови теоретичних контурів зовнішньої і внутрішньої поверхонь сопла з центральним тілом по заданому закону зміни площ (етап2) / І.В. Сергієнко, П.І. Стецюк, О.М. Литвин та інші. Заключний звіт по науково-технічному проєкту № держ. реєстрації 0119U002303. К.: Ін-т кібернетики імені В.М. Глушкова НАН України, 2020. 142 с.

3.4 ЗАСТОСУВАННЯ МОДЕЛІ UNDBE У ПОЄДНАННІ З МЕТОДОМ RALG ПРИ РОЗВ'ЯЗАННІ ЗАДАЧ РАДІОЕКОЛОГІЇ ВОДНИХ ОБ'ЄКТІВ

Т. В. Бєлих, В. П. Сизоненко

Анотація. Розглядається застосування камерної моделі перенесення забруднень у поверхневих водоймищах, яка враховує час транспортування забруднення по водойму та дисперсію забруднення в неповному об'ємі камери (UNDBE). Використовується програма RALG, що належить до модифікації r-алгоритму Шора, розробленого в Інституті кібернетики імені В.М. Глушкова. Представлено загальний опис моделі. Наводяться результати моделювання та параметричної ідентифікації на прикладах розповсюдження тритію в басейні р. Луари та стронцію-90 в Київському водосховищі. Зазначено особливості, які мали місце при проведенні параметричної ідентифікації.

Abstract. The paper considers the application of a box model of pollution transport in surface water bodies, which takes into account the time of pollution transport across the reservoir and the dispersion of pollution in the incomplete volume of the box (UNDBE). It uses the RALG program, which is a modification of the Shor's r-algorithm developed at the V.M. Glushkov Institute of Cybernetics. A general description of the model is presented. The results of modeling and parametric identification are presented on the examples of tritium distribution in the Loire River basin and strontium-90 in the Kyiv reservoir. The peculiarities of the parametric identification are noted.

3.4.1 Вступ

Наявність викидів радіоактивних забруднень у водне середовище АЕС, що працюють у штатному режимі, а також під час різних аварійних ситуацій вимагає застосування засобів прогнозування

впливу таких викидів на стан поверхневих водойм та оцінки заходів, спрямованих на зменшення впливу таких викидів.

Існує ряд розробок, що забезпечують моделювання перенесення забруднень у поверхневих водоймах, які можуть бути успішно застосовані для прогнозування поширення радіоактивних забруднень [1–3].

Маючи великі можливості моделювання процесів транспортування забруднень, вони або не мають достатньої точності, або складні і вимагають великої кількості вихідних даних і знання точних значень параметрів процесів, що протікають. Це у свою чергу викликає необхідність взяття великої кількості проб та проведення аналізів, що є окремою великою проблемою. Крім того, складні моделі вимагають значного часу реалізації на ЕОМ, через що не забезпечують можливостей параметричної ідентифікації моделі та її налаштування за даними поточних вимірів до конкретної водойми.

Для забезпечення точності прогнозування поширення забруднень у поверхневих водах в умовах обмеженості вимірювальної та обчислювальної бази, без підвищення вимог до кількості та якості вихідних даних була розроблена камерна модель перенесення забруднень, що враховує час транспортування забруднення по водойму та дисперсію забруднення в неповному об’ємі камери в процесі переносу.

Реалізована в програмних кодах модель UNDBE (UNDrikin BEaconer), яка на відміну від широко розповсюджених 1-2-3 вимірних моделей, що традиційно використовують адвекційні рівняння дисперсії (advection dispersion equation (ADE)), фактично застосовує метод агрегованої мертвої зони (aggregated dead zone (ADZ)).

Замість того, щоб моделювати концентрацію розчину безперервно як на відстані, так і в часі, модель ADZ використовує підхід «чорної шухляди» і просто розглядає концентрацію вихідного сигналу як функцію концентрації вхідного сигналу і часу.

При цьому водойма (річка, водосховище) розбивається на послідовні камери межі яких співпадають з місцями для яких необхідно зробити прогноз.

Модель у порівнянні з відомими 1-2-3 вимірними моделями вимагає значно меншої кількості початкових даних і забезпечує різке зменшення потрібного комп'ютерного часу при збереженні точності в межах наявних похибок вимірювань.

Розвиток отримав спосіб налаштування моделі за даними вимірювань, чим додатково досягається підвищення точності прогнозування. Це стало можливим саме завдяки малому часу реалізації запропонованої моделі на ЕОМ та застосуванню параметричної ідентифікації за допомогою програми RALG, що належить до модифікації r-алгоритму Шора, розробленого в Інституті кібернетики імені В.М. Глушкова.

3.4.2 Математична модель та комп'ютерна реалізація

У запропонованій камерній моделі (UNDBE), на відміну від припущення про повне і миттєве перемішування, застосовуються такі припущення:

- передбачається, що кожна порція води, що надходить у камеру разом із забрудненням, на першому етапі переміщається від втоку до витоку камери, не розбавляючись і не трансформуючись;
- по закінченню транспортування, досягнувши витоку камери, водні маси повністю моментально і рівномірно перемішуються, але лише в деякій частині об'єму камери, а забруднення трансформуються та взаємодіють із звислими намулами та донними відкладеннями.

Такі припущення призводять до моделі, що описується системою звичайних диференціальних рівнянь із запізнілим аргументом T_R (рис. 3.4.1).

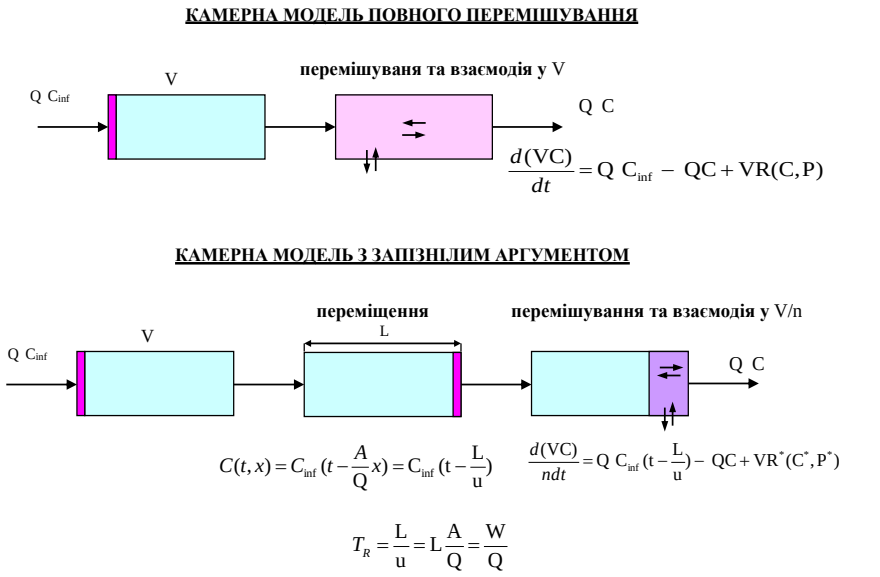


Рис. 3.4.1. Припущення звичайної камерної моделі та камерної моделі із запізнілим аргументом

Запізнілий аргумент T_R, що дорівнює часу транспортування, дозволяє врахувати той факт, що концентрацію біля витоку камери визначають водні маси, які знаходилися біля втоку в момент часу $(t - T_R)$.

З огляду на свою специфіку визначення концентрацій на витоці в момент часу t модель вимагає завдання витрат води та концентрацій на втоці в камеру в момент часу $t - T_R$.

При таких припущеннях рівняння моделі для визначення концентрації звислих намулів – S_i, концентрації забруднення у розчині – C_i, концентрації забруднення на звислих намулах – C_i^s та концентрації забруднення у шарі донних відкладень – C_i^b, для об'єму V / n, розташованого біля витоку камери, набувають вигляду

$$\frac{dV_i}{dt} = Q_{i\text{-}1} - Q_i + R_i + \sum_{j=m}^{n} Q_j^t - Q_i^w, \qquad (3.4.1)$$

$$\frac{d(V_i S_i)}{n\,dt} = Q_i\left[\tilde{S}_{i-1}(t-T_R) - S_i\right] + q_i^b - q_i^s + R_i^h - S_i Q_i^w, \quad (3.4.2)$$

$$\frac{d(V_i C_i)}{n\,dt} = Q_i\left[\tilde{C}_{i-1}(t-T_R) - C_i\right] - a_{1,2}(K_s C_i - C_i^s) -$$

$$-a_{1,3}\left(K_d C_i - C_i^b\right) - \lambda\, V_i C_i - C_i Q_i^w, \qquad (3.4.3)$$

$$\frac{d(V_i S_i C_i^s)}{n\,dt} = Q_i\left[\tilde{S}_{i\text{-}1}(t-T_R)\tilde{C}_{i\text{-}1}^s(t-T_R) - S_i C_i^s\right] + a_{1,2}(K_s C_i - C_i^s) -$$

$$-\lambda\, V_i S_i C_i^s + R_i^h C_i^h + C_i^b q_i^b - C_i^s q_i^s - Q_i^w C_i^s S_i, \quad (3.4.4)$$

$$\frac{dC_i^b}{dt} = \frac{1}{M_i^b}\left[-C_i^b\left(a_{1,3} + q_i^s + \lambda\, M_i^b\right) + C_i^s\, q_i^s + K_d a_{1,3} C_i\right], \quad (3.4.5)$$

де V_i – об'єм камери i;

Q_i – витрати води з i-ї до наступної камери;

Q_{i-1} – витрати води з попередньої камери в i-у;

$Q_j^{\,t}$ – витрати води з j притоків у камеру;

$Q_i^{\,w}$ – безповоротне водоспоживання;

R_i – різниця між величиною опадів та випарів;

$M_i^{\,b}$ – маса донних відкладень, що бере участь в обміні з розчином;

$q_i^{\,b}$, $q_i^{\,s}$ – потоки седиментації та змучування відповідно (за формулою Бійкера);

K_s, $a_{1,2}$ – параметри обміну в системі вода-суспензія;

K_d, $a_{1,3}$ – параметри обміну в системі вода-донні відкладення;

$\lambda = \ln 2 / T^*$ – постійна розпаду радіонукліду, де T^* – час напіврозпаду;

$\tilde{S}_{i-1}$, $\tilde{C}_{i-1}$, $\tilde{C}^s_{i-1}$ – концентрації звислих намулів, забруднення в розчині та забруднення на звислих намулах у втоці в камеру, які створюються завдяки притоку з попередньої камери та j бічним притокам.

Передбачається, що притоки знаходяться біля втоку камери і тоді

$$\tilde{S}_{i-1}(t-T_R) = \frac{Q_{i-1}(t-T_R)S_{i-1}(t-T_R) + \sum_j Q^t_j(t-T_R)S^t_j(t-T_R)}{Q_{i-1}(t-T_R) + \sum_j Q^t_j(t-T_R)} \quad (3.4.6)$$

$$\tilde{C}_{i-1}(t-T_R) = \frac{Q_{i-1}(t-T_R)C_{i-1}(t-T_R) + \sum_j Q^t_j(t-T_R)C^t_j(t-T_R)}{Q_{i-1}(t-T_R) + \sum_j Q^t_j(t-T_R)} \quad (3.4.7)$$

$$\tilde{C}^s_{i-1}(t-T_R) = \frac{Q_{i-1}(t-T_R)S_{i-1}(t-T_R)C^s_{i-1}(t-T_R)}{Q_{i-1}(t-T_R) + \sum_j Q^t_j(t-T_R)} + \frac{\sum_j Q^t_j(t-T_R)S^t_j(t-T_R)C^{s\,t}_j(t-T_R)}{Q_{i-1}(t-T_R) + \sum_j Q^t_j(t-T_R)}, \quad (3.4.8)$$

де Q_t, S_t, C_t – позначають витрату води, концентрації звислих намулів та забруднення в j притоках.

Час транспортування T_R залежить від швидкості течії, а також довжини камери і в загальному випадку є змінною величиною, яка визначається шляхом ідентифікації.

У рівняннях (3.4.2) – (3.4.5) концентрації є середніми не в об'ємі всієї камери, а лише середніми в $1/n$ частини об'єму, що знаходиться біля витоку. Значення $1/n$ – характеристика дисперсії, яка залежить від швидкості течії, а також об'єму камери і в загальному випадку є змінною величиною, яка визначається шляхом ідентифікації.

Модель UNDBE визначає концентрації в конкретному місці – біля витоку камери і не дає можливості розраховувати концентрації у проміжних областях камери. Тому при поділі водойми на камери витоки камер зручно розташовувати так, щоб вони збігалися з найбільш важливими для аналізу місцями водойми.

За наявності досить великих притоків, що істотно впливають на процеси водообміну в камері, розбиття на камери слід здійснювати так, щоб початок камери збігався з місцем впадання притоку, що має істотний вплив.

Як видно з рівнянь (3.4.2) – (3.4.5) запропонована модель передбачає визначення концентрацій у розчині, в суспензії, у шарі донних відкладень і концентрації самої суспензії. Отже, охоплюється широкий спектр можливих забруднень, які можуть мати місце у проточних поверхневих водоймах.

Якщо потрібно моделювати поширення забруднення, що дуже слабо взаємодіє зі звислими намулами (у цьому випадку не суттєві концентрації суспензії та забруднення в суспензії), система рівнянь (3.4.1) – (3.4.5), розв'язана щодо концентрацій, суттєво спрощується і має вигляд

$$\frac{dC_i}{dt} = \frac{n}{V_i}\left\{Q_i\tilde{C}_{i-1}\left(t-T_R\right) + a_{1,3}C_i^b - C_i \times\right.$$

$$\times \left[\frac{Q_{i\text{-}1} + R_i + \sum_j Q_j^t + (n-1)\left(Q_i + Q_i^w\right)}{n} + K_d a_{1,3} + \lambda V_i \right] \Bigg\}, \quad (3.4.9)$$

$$\frac{dC_i^b}{dt} = \frac{1}{M_i^b}\left[-C_i^b \left(a_{1,3} + \lambda\, M_i^b \right) + K_d a_{1,3} C_i \right]. \quad (3.4.10)$$

До таких забруднень належать стронцій-90 і тритій.

У системі (3.4.9) – (3.4.10) є параметри K_d, $a_{1,3}$ і n. У свою чергу

$$a_{1,3} = \frac{M_i^b}{1 + K_d \dfrac{M_i^b}{V_i}} \left(\frac{\delta_{1,3}}{\tau_{sb}} + \frac{\delta_{3,1}}{\tau_{dsb}} \right), \quad (3.4.11)$$

а $\delta_{1,3}$, $\delta_{3,1}$ – параметри, які визначають напрямок руху забруднювача при порушенні рівноваги $(K_d = C^b / C)$, які визначаються таким чином:

$$\delta_{1,3} = \left(1 - \delta_{3,1}\right) = \begin{cases} 1, & K_d C_i > C_i^b \\ 0, & K_d C_i \le C_i^b \end{cases}. \quad (3.4.12)$$

Отже, в цьому випадку при використанні запропонованої моделі знадобиться визначення чотирьох параметрів – K_d, τ_{sb}, τ_{dsb}, n.

У моделі використовуються диференціальні рівняння із запізнілим аргументом – це рівняння, що мають вигляд

$$y'(x) = f\left(x, y(x), y(x - \tau)\right). \quad (3.4.13)$$

Тут похідна рішення залежить не тільки від стану на даний момент, але також від історії в минулому.

На даний час розроблені та успішно застосовуються ефективні методи чисельного рішення таких диференціальних рівнянь [4, 5].

Комп'ютерна реалізація моделі:

- мова програмування ФОРТРАН;

- загальний обсяг – 2700 команд ФОРТРАН;
- для розв’язання диференціальних рівнянь із запізнілим аргументом була використана програма RETARD [4] з модифікацією, яка забезпечила прискорення обчислень у 1200 разів.

3.4.3 Моделювання переносу тритію в руслі р. Луари

Модель була випробувана при виконанні міжнародного проекту EMRAS (Environmental Modelling for Radiation Safety – моделювання навколишнього середовища з метою забезпечення радіаційної безпеки, Міжнародне агентство з атомної енергії) здійснювалося моделювання перенесення радіоактивного ізотопу водню – тритію в руслі р. Луари (Франція) [6].

У басейні р. Луари знаходиться п'ять атомних електростанцій (рис. 3.4.2), що включають 14 атомних реакторів, у результаті функціонування яких відбуваються викиди радіоактивного тритію у воду.

Дані вимірювань були надані DIREN Centre (Direction Régionale de l'Environnement Centre – Управління регіонального центру навколишнього середовища Франції) та EDF (Electricité de France – Електроенергетична компанія Франції).

Витрати води в семи притоках, що відповідають реальним витратам води для 1999 р., і дані про реальні викиди тритію атомними електростанціями були задані з часовою дискретністю.

Потрібно було визначити динаміку концентрацій тритію в 11 пунктах вздовж русла р. Луари (Beaulieu, Gien, Ouzouer, Orleans, Beaugency, Nouan, Tours, La Chapelle, Bertignolles, Angers, MontJean) (рис. 3.4.2) з годинною дискретністю, протягом півроку з 1 липня до 31 грудня 1999 р.

Ділянка річки довжиною 350 км описувалася шляхом завдання геометрії 486 поперечних перетинів.

Для застосування камерної моделі неповного перемішування 350-кілометрова ділянка р. Луари була розбита на 33 послідовні камери.

Границі камер розміщувалися в таких пунктах:

– початок і кінець відрізка річки, що моделюється;

– місця, де необхідно було визначити значення концентрацій;

– місця розташування водопідпірних дамб;

– місця впадання притоків;

– пункти розташування ядерних станцій (місця надходження забруднення).

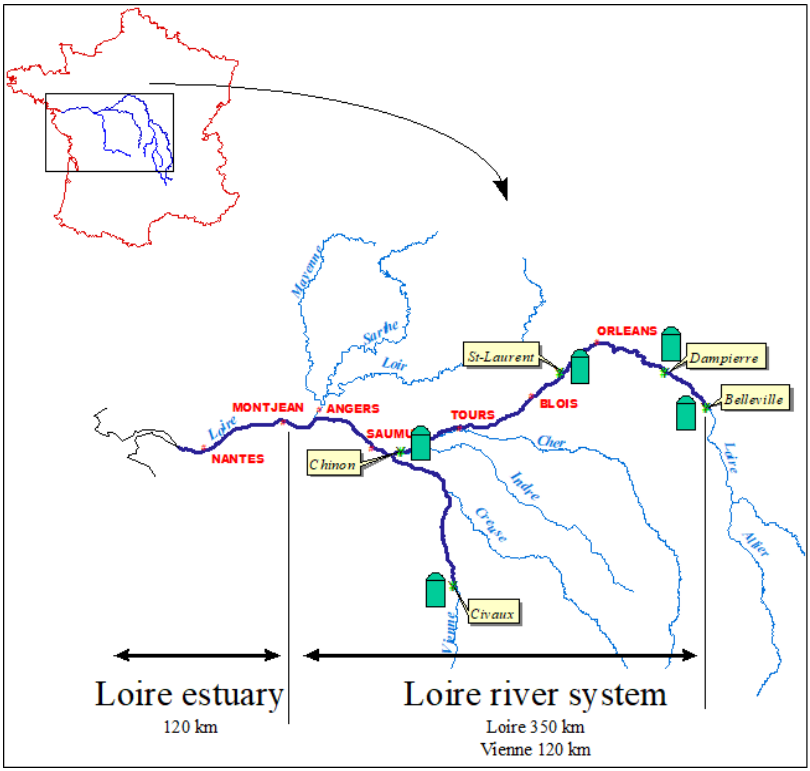


Рис. 3.4.2. Річкова система р. Луари

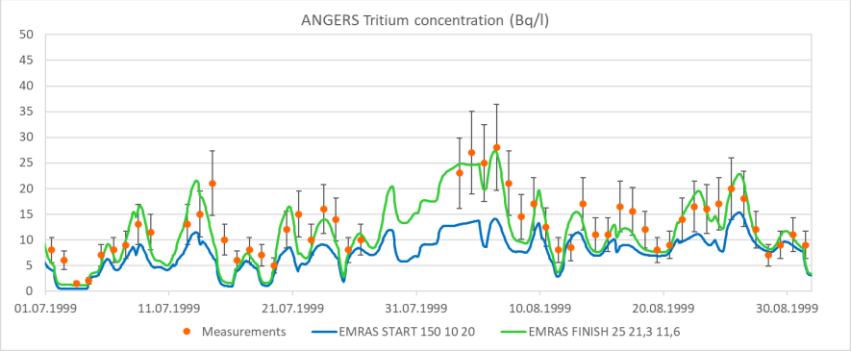
Як гідрологічна модель р. Луари для визначення поточних об’ємів камер використовувалася одновимірна стаціонарна модель [7].

Розрахунок гідрології р. Луари (33 камери 350-кілометрової ділянки, для кожної години шестимісячного інтервалу моделювання) потребує 50 хвилин часу ЕОМ з процесором Intel Core i5-9600K. Розв'язання задачі перенесення тритію за допомогою запропонованої моделі UNDBE здійснюється за 13 секунд. Оскільки гідрологічний режим річки не залежить від транспортування тритію, то розрахунок водного режиму був виконаний одноразово для всього діапазону можливих витрат води, а результати використовувалися як масив вихідних даних для багаторазових прорахунків при вирішенні задачі ідентифікації параметрів перенесення тритію, це значно прискорило визначення оптимальних значень параметрів.

Програма RALG [8, 9] використовувалася для знаходження таких значень наведеного набору параметрів, при яких забезпечується мінімальне неузгодження результатів моделювання та вимірювань.

Для оцінки неузгодженості використовувалася цільова функція – сума квадратичних відхилень наявних вимірювань та відповідних результатів моделювання.

Червоні точки на рис. 3.4.3 показують результати вимірювань концентрацій тритію у межах похибок 30 %.



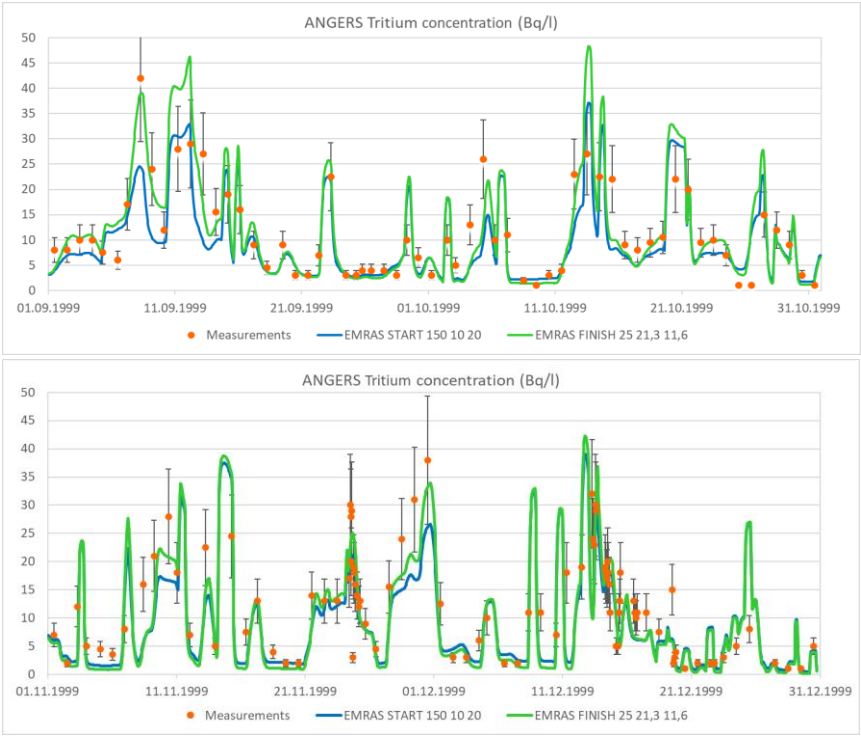


Рис. 3.4.3. Зіставлення розрахунків та вимірювань концентрацій тритію у створі ANGERS

Синя лінія на рис. 3.4.3 показує результат, отриманий при ручному доборі параметрів K_d, τ_{sb}, τ_{dsb}, n – 150, 10, 20, 11 відповідно. За таких умов цільова функція набуває значення – 0.23.

Зелена лінія на рис. 3.4.3 відображає результат ідентифікації при використанні RALG. Значення параметрів K_d, τ_{sb}, τ_{dsb} – 25, 21.3, 11.6 відповідно. Параметр n було залишено незмінним. В цьому випадку цільова функція дорівнює – 0.021.

Після того, як показано адекватність моделі в умовах багаторазових різких викидів тритію при різко змінних витратах води

у руслі Луари з щогодинною дискретністю, становить інтерес застосування моделі для каскаду дніпровських водосховищ.

3.4.4 Моделювання переносу стронцію-90 в Київському водосховищі

В результаті Чорнобильської катастрофи виникла проблема прогнозування транспортування радіонуклідів каскадом дніпровських водосховищ. Один з основних забруднювачів – стронцій-90, який при високих паводках змивається із забрудненої заплави р. Прип'ять і надходить у північну частину Київського водосховища. Для вирішення задачі перенесення стронцію-90 по водосховищам використовувалася камерна модель UNDBE, зокрема, і для Київського водосховища. Оскільки всі суттєві притоки (включаючи р. Прип'ять) знаходяться в північній частині (на втоці) Київського водосховища, а місцем, де необхідно було визначити значення концентрацій, був пункт водозабору – м. Вишгород (виток Київського водосховища), водосховище моделювалося однією камерою.

Поточні об’єми водосховища визначалися за кривими залежностей об’ємів від рівнів (h) води у водосховищах $V = n_2 h^2 + n_1 h + n_0$, де n_0, n_1, n_2 – константи [10]. Поточні дані про концентрації стронцію-90 у розчині по притоках та дані про концентрації стронцію-90 на витоку водосховища були взяті з бази даних УкрНДГМІ.

Детальну інформацію про щодобові витрати води у притоках водосховища, а також дані стосовно щодобових рівнів водосховища було отримано у Гідрометеорологічній службі України.

Враховуючи наявність щодобових вимірювань, розрахунок вівся з добовою дискретністю.

Весь розрахунок гідрології Київського водосховища та перенесення стронція-90 із добовою дискретністю для шестимісячного інтервалу моделювання потребує менше 0.03 секунди часу ЕОМ із процесором Intel Core i5-9600K.

Проведено розрахунки для 1991, 1994 та 1999 років, коли мали місце значні підйоми концентрації Sr-90 в Прип'яті. Здійснено ідентифікацію гідрологічних та токсикологічних параметрів Київського водосховища.

На рис. 3.4.4 графік, що відображає динаміку концентрації Sr-90 1991 року в районі Київської ГЕС.

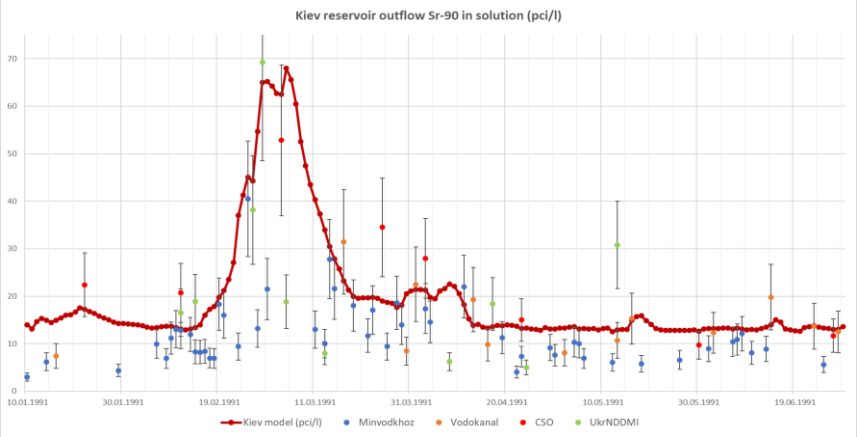


Рис. 3.4.4. Зіставлення розрахунків та вимірювань концентрацій стронцію-90 в районі м. Вишгорода 1991 р.

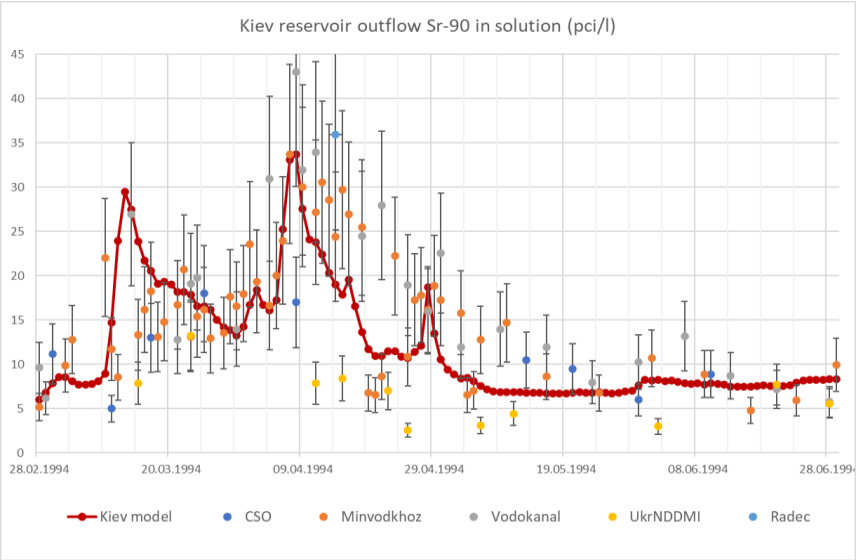


Рис. 3.4.5. Зіставлення розрахунків та вимірювань концентрацій стронцію-90 в районі м. Вишгорода 1994 р.

Різнокольорові крапки – дані вимірювань різних організацій. Великий розбіг даних вимірювань пояснюється недостатнім на той час досвідом і відсутністю інтеркалібрування. В 1994 (рис. 3.4.5), 1999 роках дані лежать кучніше. Точність вимірювань від 30 % до 50 %. На малюнках для всіх вимірювань задано 30 %.

Коричнева лінія – модельні значення. Якщо задати 40–45 % видно, що модель покриває переважну більшість вимірювань.

Застосування моделі UNDBE у поєднанні з процедурою RALG в процесі ідентифікації ілюструє рис. 3.4.6.

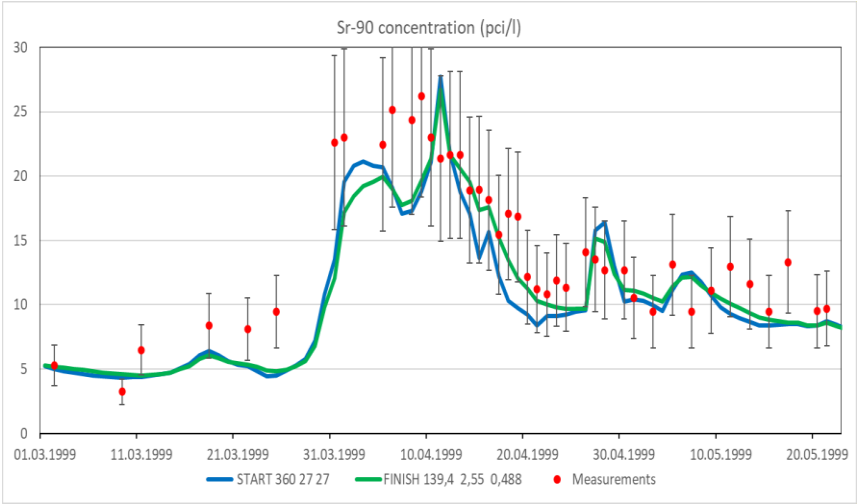


Рис. 3.4.6. Зіставлення розрахунків та вимірювань концентрацій стронцію-90 в районі м. Вишгорода 1999 р.

Синя лінія на рис. 3.4.6 показує результат, отриманий при ручному доборі параметрів K_d, τ_{sb}, τ_{dsb}, n – 360, 27, 27, 8 відповідно. Цільова функція дорівнює 0.000675.

Червоні точки показують результати вимірювань концентрацій стронцію-90 в межах похибок 30 %.

Зелена лінія на рис. 3.4.6 відображає результат ідентифікації при використанні RALG. Отримані значення параметрів K_d, τ_{sb}, τ_{dsb}, n – 139.4, 2.55, 0.488, 8 відповідно. Цільова функція після ідентифікації дорівнює 0.000539.

Варто зазначити, що:

– при обчисленні цільової функції слід виключити вимірювання, що становлять явні промахи. Інакше результат буде сильно спотворений. Так, не були використані в процесі ідентифікації вимірювання за 06.09.1999, 24.11.1999 та 19.12.1999 у пункті ANGERS на р. Луарі;

– через наявність декількох параметрів, що по-різному впливають на процес, і того, що вимірювання мають суттєві похибки, цільова функція має багато екстремальний характер. Тому починати ідентифікацію слід після того, як приблизно визначено значення параметрів. Може також знадобитися кілька різних прорахунків програмою RALG з різними початковими значеннями параметрів;

– виходячи з практики, початковий крок процедури RALG краще задавати приблизно 0.1 значення мінімального параметра.

3.4.5 Висновки

Застосування моделі UNDBE у поєднанні з процедурою RALG при вирішенні задач радіоекології водних об'єктів забезпечує:

– швидкий розрахунок за збереження точності моделювання без значного ускладнення моделі;

– збереження мінімальних вимог до кількості вихідних даних, що у поєднанні зі швидкодією, дає можливість застосування як системи аварійного реагування;

– підвищення точності моделювання за рахунок параметричної ідентифікації;

– можливість швидкого та точного налаштування для конкретного водного об'єкта;

– оцінку гідрологічних та токсикологічних параметрів водойм;

– визначення можливого впливу водоохоронних заходів.

Модель має великий потенціал розвитку, зокрема застосування для різних водних об’єктів і різних типів забруднення, зокрема для всього каскаду дніпровських водосховищ.

Список літератури

1. Rauch W., Henze M., Koncsos L., Reichert P., Shanahan P., Vanrolleghem P. River water quality modeling: I. state of the art. Water Science and Technology. 1998. Vol. 38, No. 11. P. 237–244.

2. Shanahan P., Henze M., Koncsos L., Rauch W., Reichert P., Vanrolleghem P. River water quality modeling: II. problems of the art. Water Science and Technology. 1998. Vol. 38, No. 11. P. 245–252.
3. Orlob G.T (Ed.), Mathematical Modeling of Water Quality: Streams, Lakes, and Reservoirs, International Series. On Applied Systems Analysis, IIASA, Pitman Press. 1983. No. 12. 518 p.
4. Хайрер Э., Нерсетт С., Ваннер Г. Решение обыкновенных дифференциальных уравнений. Нежесткие задачи. М.: Мир, 1990. 593 с.
5. Хайрер Э., Ваннер Г. Решение обыкновенных дифференциальных уравнений: Жесткие и дифференциальные алгебраические задачи. М.: Мир, 1999. 685 с.
6. Testing of Models for Predicting the Behaviour of Radionuclides in Freshwater Systems and Coastal Areas. http://www-ns.iaea.org/downloads/rw/projects/emras/final-reports/aquatic-tecdoc-final.pdf (звернення: 29.11.2022)
7. Караушев А.В. Речная гидравлика. Л.: Гидрометеоиздат, 1969. 416 с.
8. Шор Н.З. Методы минимизации не дифференцируемых функций и их приложения. Киев : Наук. думка, 1979. 200 с.
9. Шор Н.З., Стеценко С.И. Квадратичные экстремальные задачи и не дифференцируемая оптимизация. Киев: Наук. думка, 1989. 208 с.
10. Каганер М.С. Гидрометеорологический режим озёр и водохранилищ СССР. Каскад днепровских водохранилищ. Л.: Гидрометеоиздат, 1976. 348 с.
11. Keith Richardson, Paul Anthony Carling. The hydraulics of a straight bedrock channel: Insights from solute dispersion studies. Geomorphology 82 (2006) 98–125pp.
12. Steve WALLIS. On the Numerical Solution of the ADZ Model, PUBLS. INST. GEOPHYS. POL. ACAD. SC., E-7 (401), 2007.

ДОДАТОК А. Ratfor Програма SolverA

```
# Програма SolverA знаходить мінімальні за сумарною
# вартістю значення пропускних спроможностей дуг
# орієнтованої мережі, які гарантують виконання всіх
# вимог на передачу заданих обсягів потоку в мережі
# при можливих одиничних відмовах  мережі
# (див. Модель А (1.1.1) – (1.1.5))
# Програма реалізована на основі схем декомпозиції
# за змінними та обмеженнями. Негладкі задачі
# вирішуються r-алгоритмом.

# Максимальні розміри мережі і вимог на пересилку потоків
define MaxArc 50  # Максимальне число дуг в мережі N(V,A)
define MaxVert 25 # Максимальне число вершин мережі N(V,A)
# Максимальне число поставок MaxVert*(MaxVert-1)
define MaxDemand 600
# MaxVert + 1, використовується для адресації вимог
define MaxVert1 26
# Максимально число поломок (включаючи і нульову)
define MaxTr 51
# MaxTr + 1, використовується для адресації поломок
define MaxTr1 52
define MaxArcTr 200 # Максимальне число дуг у всіх відмовах
# Використовувані номери каналів для роботи з файлами
define irdr1 1  # читається мережа
define irdr2 2  # читаються вимоги для потоків в мережі
define irdr3 3  # читаються поломки в мережі
define iprint 6 # друкується протокол роботи програми
# Постійні параметри r-алгоритму
define alp 4.d0 # коефіцієнт розтягу простору
# число одновимірних спусків без збільшення кроку
define nh 3
# коефіцієнт збільшення кроку якщо перевищено nh
define q2 1.1d0
define maxitn 500 # максимальне число ітерацій
define epsg 1.d-4 # точність зупинки по нормі субградієнта

# Опис даних задачі
```

```
# всі дані описані так
implicit real*8 (a-h,o-z),integer*2(i-n)

# Блок /NetVA/: орієнтована (directed) мережа N(V,A)
common/NetVA/nVert,nArc,iv(MaxArc),jv(MaxArc),
             cost(MaxArc),y0(MaxArc),yup(MaxArc)
# nVert – кількість вершин; nArc - кількість дуг;
# Для k-ої дуги (k=1,...,nArc) задані:
# iv(k) - початкова вершина,
# jv(k) - кінцева вершина,
# cost(k) - вартість одиничної пропускної спроможності
# y0(k) - наявне значення пропускної спроможності

# Блок /Demand/: вимоги на пересилку обсягів потоку в
# мережі N(V,A)
common/Demand/is(MaxVert1),isr(MaxDemand),dsr(MaxDemand)
# is(*) - масив покажчиків на початок поставок з вершин
# is(i) - містить покажчик на початок поставок з вершини i
#         в наявний список вершини в масивах isr(*), dsr(*)
# is(nVert+1) - містить покажчик на вільне місце в isr(*) и
# dsr(*)
#         В інші вершини в масиві isr(*), dsr(*)
# isr(*)  містить індекси вершин, куди постачати
# dsr(*)  містить самі обсяги поставок

# Блок /Troubl/: Поломки в мережі (перша відповідає
# звичайному режиму функціонування мережі,
# тобто нульовій поломці)
common/Troubl/nTr,nt(MaxTr1),narct(MaxArcTr),arcmut(MaxArcT
r)
# nTr       - кількість поломок в мережі (включаючи
# нульову)
# nt(nTr+1) - масив покажчиків на початок поломок
# narct(*)  - містить номери дуг для поломок
# arcmu(*)  - містить коефіцієнти зменшення пропускної
# спроможності дуг для конкретної поломки

common/Data/Penalt,Zup    # блок для установки параметрів
                          # для додаткових змінних z_ijt
# Penalt - значення штрафу для змінних z_ij
# (обчислюється...)
```

```
# Zup    - верхня межа на змінні z_ij (обчислюється ....)
# Ще два масиви:  yopt(*) - оптимальні значення
# пропускних спроможностей дуг, що додаються
dimension yopt(MaxArc),y(MaxArc)
call prtime(0)
call readDataA  #Читання і підготовка даних

# Встановити штраф і верхню межу для змінних z_ijt
# (блок /Data/)
Penalt=1.d0;
for(i=1; i<=nArc; i=i+1) {
   Penalt=Penalt+cost(i)
}
Zup=0.d0;
ind=1; ist=is(nVert+1)-1
for(i=ind; i<=ist; i=i+1) {
    Zup=Zup+dsr(i)
}
write (iprint,'(/3x,a)')' Для додаткових змінних обрані:'
write (iprint,'(13x,a,3x,g17.5)')' Penalty=',Penalt
write (iprint,'(13x,a,3x,g17.5)')' Z_upper=',Zup

call prtime(1)

ny=nArc
do i=1,ny
   y(i)=100.d0
# Інші вхідні параметри r-алгоритму
h0=1.d0; epsy=1.d-5; intp=1;
call CoordA(ny,y,h0,epsy,intp,fopt,yopt,itn,istop)

write (iprint,'(/3x,a)')' Оптимізацію закінчено:'
write (iprint,'(13x,a,3x,f15.5)')' Fopt = ',fopt
write (iprint,'(13x,a,8x,i5)')' Iteration = ',itn
write (iprint,'(13x,a,12x,i5)')' Istop = ',istop
write (iprint,'(/3x,a)')' Знайдено наступний розвязок:'
write(iprint,'(2x,3(4x,a),3x,a)')'N','i','j','yopt(i,j)'
fy=0.d0
do j=1,ny{
  write(iprint,'(2x,3i5,2(2x,f10.2))')j,iv(j),jv(j),yopt(j)
  fy=fy+cost(j)*yopt(j)
```

```
}
write (iprint,'(/3x,a,2(3x,f12.1),5x,i3)')' Разом: Fopt
F(yopt)',fopt,fy
call prtime(2)
stop
end

# Програма читання і перевірки вихідних даних задачі
# Підготовка блоків /NetVA/, /Demand/ і /Troubl/
subroutine readDataA
implicit real*8 (a-h,o-z),integer*2(i-n)

common/NetVA/nVert,nArc,iv(MaxArc),jv(MaxArc),
             cost(MaxArc),y0(MaxArc),yup(MaxArc)
common/Demand/is(MaxVert1),isr(MaxDemand),dsr(MaxDemand)
common/Troubl/nTr,nt(MaxTr1),narct(MaxArcTr),arcmut(MaxArcT
r)

# Робочі масиви: відповідають даним з файлу вимог
dimension is1(MaxDemand),ir1(MaxDemand),dsr1(MaxDemand)
logical Fizi

# Прочитати блок /NetVA/ і перевірити максимально можливі
# межі
read(irdr1,*)nVert,nArc
if(nVert>MaxVert) {
  write(iprint,*)' Кількість вершин перевищує максимально
допустиму'
  i1=MaxVert
  write(iprint,*)' nVert=',nVert,' MaxVert=',i1
  stop
}
if(nArc>MaxArc) {
  write(iprint,*)' Кількість дуг перевищує максимально
допустиму, тобто'
  i1=MaxArc
  write(iprint,*)' nArc=',nArc,' MaxArc=',i1
  stop
}
write(iprint,'(/10x,a)')' Вихідні дані для моделі A'
```

```
write(iprint,'(/3x,2(a,i5))')' Структура мережі:
nVert=',nVert,' nArc=',nArc
write(iprint,'(2x,3(4x,a),6x,a,5x,a)')'N','i','j','c(i,j)',
'y0(i,j)'
do i=1,nArc{
   read(irdr1,*)iv(i),jv(i),cost(i),y0(i)
write(iprint,'(2x,3i5,2(2x,f10.2))')i,iv(i),jv(i),cost(i),y
0(i)
   if(iv(i)<=nvert & iv(i)>=1 & jv(i)<=nvert & jv(i)>=1 &
     cost(i)>=0.d0 & y0(i)>=0.d0) next
   write(iprint,*)' Error: щось не так в попередньому
рядку...'
   stop
}

# Прочитати файл вимог і перевірити максимальні межі
read(irdr2,*)ndem1
if(ndem1>MaxDemand) {
  write(iprint,*)' Занадто багато вимог на пересилку
потоків '
  i1=MaxDemand
  write(iprint,*)' Кількість вимог ',ndem1,' MaxDemand=',i1
  stop
}
write(iprint,'(/3x,a)')' Вимоги на пересилку потоків в
мережі '
write(iprint,'(2x,3(4x,a),6x,a)')'N','s','r','d(s,r)'
do i=1,ndem1{
   read(irdr2,*)is1(i),ir1(i),dsr1(i)
   write(iprint,'(2x,3i5,2x,f10.2)')i,is1(i),ir1(i),dsr1(i)
   if(is1(i)<=nvert & is1(i)>=1 & ir1(i)<=nvert & ir1(i)>=1
&
     dsr1(i)>=0.d0) next
   write(iprint,*)' Error: щось не так в попередньому рядку
...'
   }
is(1)=1; nelms=0;
do i=1,nVert{
   do j=1,ndem1{
      if(is1(j)==i) {
        nelms=nelms+1
```

```
        isr(nelms)=ir1(j)
        dsr(nelms)=dsr1(j)
      }
   }
   is(i+1)=nelms+1
}

# Прочитати і заповнити масив поломок
nTr=1; nt(1)=1; nt(2)=1; nelms1=0;
write(iprint,'(/3x,a)')' Поломки в мережі '
write(iprint,'(2x,3(4x,a),6x,a)')'N','i','j','mu(i,j)'
repeat{
   read(irdr3,*)ntt
   if(ntt==0) break
   nTr=nTr+1
   if(nTr>MaxTr) {
      write(iprint,*)' Занадто багато поломок в мережі '
      i1=MaxTr
      write(iprint,*)' Всього ',nTr,' MaxTr=',i1
      stop
   }
   Fizi=.true.
   for(i=1; i<=ntt; i=i+1){
      nelms1=nelms1+1
      if(nelms1>MaxArcTr) {
         write(iprint,*)' Занадто багато дуг в масиві
поломок'
         i1=MaxArcTr
         write(iprint,*)' Всього ',nelms1,' MaxTr=',i1
         stop
      }
      read(irdr3,*)iv1,jv1,arcmut(nelms1)
      nTr1=nTr-1
      if(Fizi) {

write(iprint,'(2x,3i5,2x,f10.2)')nTr1,iv1,jv1,arcmut(nelms1
)
        Fizi=.false.
      }
      else {
```

```
write(iprint,'(7x,2i5,2x,f10.2)')iv1,jv1,arcmut(nelms1)
      }
      narct(nelms1)=0
      for(j=1; j<=nArc; j=j+1){
         if(iv(j)!=iv1) next
         if(jv(j)!=jv1) next
         narct(nelms1)=j
         break
      }
      if(narct(nelms1)!=0 & arcmut(nelms1)>=0.d0 &
         arcmut(nelms1)<=1.d0 ) next
      write(iprint,*) 'Error: Невірно вказана остання
поломка'
      stop
   }
   nt(nTr+1)=nelms1+1
}
return
end

# Підпрограма CoordA, r-алгоритм вирішує координуючу
# задачу мінімізації негладкої опуклої функції
# для схеми декомпозиції за змінними
# Вхідні параметри:
# n - розмірність простору змінних
# h0 - початковий крок
# epsx - критерій зупинки по аргументу
# x(n) - початкова точка
# Вихідні параметри:
# itn - число витрачених ітерацій
# istop - код зупинки (2-по epsg, 3-по epsx,
# 4-по числу ітерацій, 5 - не знайдений мінімум у напрямку)
# xr (n) - знайдена точка мінімуму функції
# fr - значення функції в точці мінімуму
subroutine CoordA(n,x,h0,epsx,intp,fr,xr,itn,istop)
implicit real*8(a-h,o-z),integer*2(i-n)
dimension x(n),xr(n)
# Робочі масиви: b(n,n) - матриця оберненого перетворення
# простору
# g(n),g1(n),g2(n) – проміжні вектори
```

```
dimension b(MaxArc,MaxArc),g(MaxArc),g1(MaxArc),g2(MaxArc)
# Установка нуля і початкових параметрів
dzero=1.d-20; w=1./alp-1.
hs=h0
itn=0; lp=itn+intp
do i=1,n{
   do j=1,n
      b(j,i)=0.d0
   b(i,i)=1.d0
   }
call FGCoord(x,n,f,g)
fr=f
do i=1,n{
   g1(i)=g(i); xr(i)=x(i)
}
nls=0; nlsa=0
write(iprint,3000)
write(iprint,3100) itn,f,fr,nlsa,nls
# Основне тіло програми, що реалізує алгоритм
for(itn=1; itn<=maxitn; itn=itn+1){
# Критерій зупинки по нормі градієнта
   dg=0.d0
   do i=1,n
      dg=dg+g(i)*g(i)
   istop=2
   if(dsqrt(dg)<=epsg) break
# Обчислити субградієнт в перетвореному просторі
   do i=1,n{
      d=0.d0
      do j=1,n
         d=d+b(j,i)*g(j)
      g2(i)=d
   }
# Обчислити і нормувати різницю субградієнтів
   dg=0.d0
   do i=1,n{
      g(i)=g2(i)-g1(i)
      dg=dg+g(i)*g(i)
   }
   dg=dsqrt(dg)
   if(dg>dzero) {
```

```
      dg=1.d0/dg
      do i=1,n
         g(i)=dg*g(i)
   }
# Обчислити нормований субградієнт в перетвореному просторі
   d=0.d0
   do i=1,n
      d=d+g(i)*g2(i)
   d=w*d
   d1=0.d0
   do i=1,n{
      g1(i)=g2(i)+d*g(i)
      d1=d1+g1(i)**2
   }
   d1=1./dsqrt(d1)
   do i=1,n
      g2(i)=d1*g1(i)
# Перерахувати матрицю B
   do i=1,n{
      d=0.d0
      do j=1,n
         d=d+b(i,j)*g(j)
      d=w*d
      do j=1,n
         b(i,j)=b(i,j)+d*g(j)
   }
# Обчислити напрямок руху
   dg=0.d0
   do i=1,n{
      d=0.d0
      do j=1,n
         d=d+b(i,j)*g2(j)
      g(i)=d
      dg=dg+d*d
   }
   dg=dsqrt(dg)
# Одновимірний спуск у напрямку
   ls=0; dx=0.d0; istop=5
   20 continue
     ls=ls+1
     dx=dx+hs*dg
```

```
      do i=1,n
         x(i)=x(i)-hs*g(i)
      call FGCoord(x,n,f,g2)
      if(f<fr) {
        fr=f
        do i=1,n
           xr(i)=dabs(x(i))
      }
      d=0.d0
      do i=1,n
         d=d+g(i)*g2(i)
      if(ls>500) break  # Останов: istop = 5
      if(ls>nh) hs=hs*q2
   if(d>0.d0) go to 20
   nls=nls+ls; nlsa=nlsa+ls
   if(itn==lp) {
     write(iprint,3100) itn,f,fr,nlsa,nls
      lp=lp+intp; nlsa=0
   }
   do i=1,n
      g(i)=g2(i)
   istop=3
   sum=0.d0
   do i=1,n
      sum=sum+x(i)*x(i)+1.d0
   sum=dsqrt(sum)
   if(dabs(dx/sum)<epsx) break
   istop=4
}
if(istop==4) return
if(itn==lp) write(iprint,3100) itn,f,fr,nlsa,nls
return
3000 format(/10x,' Протокол процесу негладкої оптимізації
'/2x,
             '  Itn.',7x,'..f(x)..',13x,'...f(x_r)...',
             5x,'LS',4x,'LSa')
3100 format(2x,i5,2(2x,1pd18.10),3(2x,i5))
end
# Програма FGCoord - обчислює значення функції і
# субградієнта для негладкої функції яка відповідає
# декомпозиції за змінними
```

```
subroutine FGCoord(y,ny,fy,gy)
implicit real*8 (a-h,o-z),integer*2(i-n)
# Загальні блоки даних
common/NetVA/nVert,nArc,iv(MaxArc),jv(MaxArc),
             cost(MaxArc),y0(MaxArc),yup(MaxArc)
common/Troubl/nTr,nt(MaxTr1),narct(MaxArcTr),arcmut(MaxArcT
r)
dimension y(ny),gy(ny)
# Робочі масиви
dimension xmu1(MaxArc),y1(MaxArc),u(MaxArc),ur(MaxArc)
fy=0.d0
do i=1,ny{
   gy(i)=cost(i)
   fy=fy+cost(i)*dabs(y(i))
}
nu=nArc;
for(it=1;it<=nTr;it=it+1){
   do i=1,nu
      xmu1(i)=1.d0
   ind=nt(it); ist=nt(it+1)-1;
   for(i=ind; i<=ist; i=i+1){
      xmu1(narct(i))=arcmut(i)
   }
   do i=1,nu{
      u(i)=0.d0  # старт з нульової точки
      # значення правої частини в підзадачі
      y1(i)=xmu1(i)*(y0(i)+dabs(y(i)))
   }
   # Параметри внутрішнього r-алгоритму
   h0=1.d0; epsu=1.d-5; intp=-1;
   call DualA(nu,u,h0,epsu,intp,fr,ur,itn,istop,y1)
   if(istop>3) {
      write (iprint,*)' *** Warning: Не розвязано внутрішню
підзадачу'
      write (iprint,*)'   t =',it, '    istop= ',istop
   }
   fy=fy+fr
 # write (iprint,*)'   t =',it, '    fr= ',fr
   do i=1,nu
      gy(i)=gy(i)-xmu1(i)*ur(i)
}
```

```
do i=1,nu
   # урахування заміни на модуль
   if(y(i)<0.d0) gy(i)=-gy(i)
return
end
# Підпрограма DualA, r-алгоритм максимізує увігнуту
# негладку функцію для схеми декомпозиції за обмеженнями
# Вхідні параметри:
# n - розмірність простору змінних
# h0 - початковий крок
# epsx - критерій зупинки по аргументу
# x(n) - початкова точка
# Вихідні параметри:
# itn - число витрачених ітерацій
# istop - код зупинки (2-по epsg, 3-по epsx, 4-по числу
ітерацій,
# 5 - не знайдений мінімум у напрямку, 10 - по
несумісності)
# xr (n) - знайдена точка максимуму функції
# fr - значення функції в точці максимуму
subroutine DualA(n,x,h0,epsx,intp,fr,xr,itn,istop,y)
implicit real*8(a-h,o-z),integer*2(i-n)
dimension x(n),xr(n),y(n)
# Робочі масиви:
# b(n,n) - матриця оберненого перетворення простору
# g(n),g1(n),g2(n) - зберігання проміжних векторів
dimension b(MaxArc,MaxArc),g(MaxArc),g1(MaxArc),g2(MaxArc)
# Установка нуля і початкових параметрів
dzero=1.d-20; w=1./alp-1.
hs=h0
itn=0; lp=itn+intp
do i=1,n{
   do j=1,n
      b(j,i)=0.d0
   b(i,i)=1.d0
   }
call FGDual(x,n,f,g,y)
fr=f
do i=1,n{
   g1(i)=g(i); xr(i)=dabs(x(i))
}
```

```
nls=0; nlsa=0
if(intp>0) {
  write(iprint,3000)
  write(iprint,3100) itn,f,fr,nlsa,nls
  }
# Основне тіло програми, що реалізує алгоритм
for(itn=1; itn<=maxitn; itn=itn+1){
# Критерій зупинки по нормі градієнта
   dg=0.d0
   do i=1,n
      dg=dg+g(i)*g(i)
   istop=2
   if(dsqrt(dg)<=epsg) break
# Обчислити субградієнт в перетвореному просторі
   do i=1,n{
      d=0.d0
      do j=1,n
         d=d+b(j,i)*g(j)
      g2(i)=d
   }
# Обчислити і нормувати різницю субградієнтов
   dg=0.d0
   do i=1,n{
      g(i)=g2(i)-g1(i)
      dg=dg+g(i)*g(i)
   }
   dg=dsqrt(dg)
   if(dg>dzero) {
     dg=1.d0/dg
     do i=1,n
        g(i)=dg*g(i)
   }
# Обчислити нормований субградієнт в перетвореному просторі
   d=0.d0
   do i=1,n
      d=d+g(i)*g2(i)
   d=w*d
   d1=0.d0
   do i=1,n{
      g1(i)=g2(i)+d*g(i)
      d1=d1+g1(i)**2
```

```
   }
   d1=1./dsqrt(d1)
   do i=1,n
      g2(i)=d1*g1(i)
# Перерахувати матрицю B
   do i=1,n{
      d=0.d0
      do j=1,n
         d=d+b(i,j)*g(j)
      d=w*d
      do j=1,n
         b(i,j)=b(i,j)+d*g(j)
   }
# Обчислити напрямок руху
   dg=0.d0
   do i=1,n{
      d=0.d0
      do j=1,n
         d=d+b(i,j)*g2(j)
      g(i)=d
      dg=dg+d*d
   }
   dg=dsqrt(dg)
# Одновимірний спуск у напрямку
   ls=0; dx=0.d0; istop=5
   20 continue
     ls=ls+1
     dx=dx+hs*dg
     do i=1,n
        x(i)=x(i)+hs*g(i)
     call FGDual(x,n,f,g2,y)
     if(f>fr) {
       fr=f
       do i=1,n
          xr(i)=dabs(x(i))
     }
     d=0.d0
     do i=1,n
        d=d+g(i)*g2(i)
     if(ls>500) break  # Останов: istop = 5
     if(ls>nh) hs=hs*q2
```

```
   if(d>0.d0) go to 20
   nls=nls+ls; nlsa=nlsa+ls
   if(itn==lp & intp>0) {
     write(iprint,3100) itn,f,fr,nlsa,nls
      lp=lp+intp; nlsa=0
   }
   do i=1,n
      g(i)=g2(i)
   istop=3
   sum=0.d0
   do i=1,n
      sum=sum+x(i)*x(i)+1.d0
   sum=dsqrt(sum)
   if(dabs(dx/sum)<epsx) break
   istop=4
}
if(istop==4) return
if(intp>0) write(iprint,3100) itn,f,fr,nlsa,nls
return
3000 format(/10x,' Протокол процесу негладкої оптимізації
'/2x,
             '  Itn.',7x,'..f(x)..',13x,'...f(x_r)...',
             5x,'LS',4x,'LSa')
3100 format(2x,i5,2(2x,1pd18.10),3(2x,i5))
end
# Підпрограма FGDual обчислює функцію і субградієнт для
# Dual
subroutine FGDual(u,nu,fu,g,y1)
implicit real*8 (a-h,o-z),integer*2(i-n)
# Загальні блоки даних
common/NetVA/nVert,nArc,iv(MaxArc),jv(MaxArc),
             cost(MaxArc),y0(MaxArc),yup(MaxArc)
common/Demand/is(MaxVert1),isr(MaxDemand),dsr(MaxDemand)
common/Data/Penalt,Zup
# Вхідні параметри:
# nu - кількість множників Лагранжа (дорівнює числу дуг)
# u(nu) - вектор множників Лагранжа
# y1(nu) - вектор правих частин зв'язуючих обмежень
# Вихідні параметри:
# fu - значення функції координуючої задачі,
# g(nu) - субградієнт координуючої задачі,
```

```
dimension u(nu),g(nu),y1(nu)
# Робочі масиви
# cu(MaxArc) - вектор множників Лагранжа, u_i = dabs(z_i)
# pot(MaxVert) - вектор потенціалів для знаходження
найкоротшого шляху
# ipred(MaxVert) - вектор посилань на дуги для розшифровки
найкоротших шляхів
# використовується при зупинці алгоритму найкоротших шляхів
dimension cu(MaxArc),pot(MaxVert),ipred(MaxVert)
logical fizi
# Присвоїти початкові значення функції і субградієнту з
# урахуванням заміни на модуль, що відповідає
# cu(i) = dabs(u(i))
fu=0.d0;
do i=1,nu{
   cu(i)=dabs(u(i)) # запамятати заміну на модуль
   g(i)=-y1(i)      # урахування вектора правих частин
   fu=fu+g(i)*cu(i)
}
# константи для найкоротших шляхів
dmax=1.d10; dmax1=dmax-1.d0; deps=1.d-8;
for(i=1; i<=nVert; i=i+1) {
   # покажчики кому постачати з вершини i
   ind=is(i); ist=is(i+1)-1
   # якщо нікому, то нічого не робити
   if(ind>ist) next
   for(j=1; j<=nVert; j=j+1){    # ініціалізація масивів,
      pot(j)=dmax; ipred(j)=-1;  # потенціалів і посилань
      }
   # відкрити початок (вершина i)
   pot(i)=0.d0; ipred(i)=0;
   # цикл перерахунку потенціалів
   repeat{
      # для алгоритму найкоротших шляхів
      fizi=.true.
      for(j=1; j<=narc; j=j+1){
         if(pot(iv(j))>dmax1) next
         a1=pot(iv(j))+cu(j)
         if(a1<pot(jv(j))-deps) {
           # запамятати найкращий потенціал
           pot(jv(j))=a1;
```

```
            # i номер дуги, звідки він прийшов
            ipred(jv(j))=j;
            fizi=.false.
          }
       }
       # Зупинка, якщо нічого покращувати
       if(fizi) break
    }
    for(j=ind; j<=ist; j=j+1){
       i1=isr(j);
       if(ipred(i1)==-1) {
         write(iprint,*) ' Error 1: не знайдено найкоротший
шлях з '
         write(iprint,*) '          вершини',i,' до
вершини',i1
         stop
       }
       fu=fu+pot(i1)*dsr(j)   # скоригувати fu
       repeat {               # скоригувати субградієнт
          i0=ipred(i1)
          g(i0)=g(i0)+dsr(j)
          i1=iv(i0)
          # розшифровку найкоротшого шляху закінчено
          if(i1==i) break
       }
    }
}
# Лінійний штраф (\sum_{i,j}C_ij}+1)*Z_ijt,
# 0 <= z_ijt <= \sum_nd dem(i)
for(i=1; i<=nu; i=i+1) {
   zij=Penalt-cu(i)
   if(zij>=0.d0) next
   fu=fu+zij*Zup
   g(i)=g(i)-Zup
}
for(i=1; i<=nu; i=i+1) {
   # урахування неактивних обмежень
   if(cu(i)<1.d-8 & g(i)<1.d-8) g(i)=0.d0
   # урахування заміни змінних на їх модуль
   if(u(i)<0.d0) g(i)=-g(i)
}
```

```
return
end
subroutine prtime(icode)
common /time/ih0,im0,is0,iss0
integer*2 ih0,im0,is0,iss0
integer*2 ih,im,is,iss
integer*2 i1,i2,i3,i4
integer values(8)
if(icode==0){
  call date_and_time(VALUES=values)
  ih0 = values(5)
  im0 = values(6)
  is0 = values(7)
  iss0 = values(8)
  return
  }
call date_and_time(VALUES=values)
ih = values(5)
im = values(6)
is = values(7)
iss = values(8)
i1=ih-ih0
if(im<im0) {
  im=im+60
  i1=i1-1
  }
i2=im-im0
if(is<is0) {
  is=is+60
  i2=i2-1
  }
i3=is-is0
if(iss<iss0) {
  iss=iss+1000
  i3=i3-1
  }
i4=iss-iss0
write (*,'(a,3(i2,1h.),i3)')' time ',i1,i2,i3,i4
return
end
```

```
############################################################
# Вхідні файли для тестового прикладу (див. підрозділ 1.1)
############################################################
# fort.1
############################################################
    6    16
    1    2       1.50      0.0
    1    3       1.00      0.0
    1    4       1.00      0.0
    2    4       1.00      0.0
    2    5       1.00      0.0
    3    4       1.00      0.0
    4    5       1.00      0.0
    3    6       1.00      0.0
    2    1       1.50      0.0
    3    1       1.00      0.0
    4    1       1.00      0.0
    4    2       1.00      0.0
    5    2       1.00      0.0
    4    3       1.00      0.0
    5    4       1.00      0.0
    6    3       1.00      0.0

############################################################
# fort.2
############################################################
    30
    1    2       10.00
    2    1       10.00
    1    3       10.00
    3    1       10.00
    1    4       10.00
    4    1       10.00
    1    5       10.00
    5    1       10.00
    1    6       10.00
    6    1       10.00
    2    3       10.00
    3    2       10.00
    2    4       10.00
```

```
    4     2        10.00
    2     5        10.00
    5     2        10.00
    2     6        10.00
    6     2        10.00
    3     4        10.00
    4     3        10.00
    3     5        10.00
    5     3        10.00
    3     6        10.00
    6     3        10.00
    4     5        10.00
    5     4        10.00
    4     6        10.00
    6     4        10.00
    5     6        10.00
    6     5        10.00

#############################################################
# fort.3 без поломок a)
#############################################################
    0

#############################################################
# fort.3 поломка b)
#############################################################
   1
   1     2    0.d0
   1
   1     3    0.d0
   1
   1     4    0.d0
   1
   2     4    0.d0
   1
   2     5    0.d0
   1
   3     4    0.d0
   1
   4     5    0.d0
   1
```

```
   2    1   0.d0
   1
   3    1   0.d0
   1
   4    1   0.d0
   1
   4    2   0.d0
   1
   5    2   0.d0
   1
   4    3   0.d0
   1
   5    4   0.d0
  0

#############################################################
# fort.3 поломка c)
#############################################################
   4
   1    2   0.d0
   2    1   0.d0
   2    4   0.d0
   4    2   0.d0
   0

#############################################################
# fort.3 поломка f)
#############################################################
   1
   1    2   0.5d0
   1
   1    3   0.5d0
   1
   1    4   0.5d0
   1
   2    4   0.5d0
   1
   2    5   0.5d0
   1
   3    4   0.5d0
   1
```

```
4     5    0.5d0
1
2     1    0.5d0
1
3     1    0.5d0
1
4     1    0.5d0
1
4     2    0.5d0
1
5     2    0.5d0
1
4     3    0.5d0
1
5     4    0.5d0
1
6     3    0.5d0
1
3     6    0.5d0
0
```

ДОДАТОК Б. k-Вершинний цикл: AMPL-реалізація задач

```
AMPL-код задач (1.3.1)-(1.3.10) та (1.3.11)-(1.3.18)
(на прикладі графа st70.tsp)
*********************************************************

param n>=5; # кількість вершин в графі
param s>=1<=n; # фіксована вершина (початок циклу)
param k>=1<=(n-1) default n-1; # кількість вершин циклу

set NODES := 1..n; # вершини та дуги повного графа
set ARCS within (NODES cross NODES) := {i in NODES, j in
NODES: i!=j};
# координати вершин
param xcoord{NODES}; param ycoord{NODES};
# довжини дуг (евклідова відстань, округлена)
param d{ARCS} >= 0;

# Невідомі (змінні)
var x{ARCS} binary; var y{NODES} binary; # булеві
var z{ARCS} >=0; # неперервні
var u{NODES} >= 1 <= k integer; # цілочислові

# Мінімізувати довжину циклу:
minimize dk_min: sum{(i,j) in ARCS} d[i,j] * x[i,j];
subject to # за обмежень
con2 {i in NODES}:  # одноразовий вхід у вершину i
sum{j in NODES: (i,j) in ARCS } x[i,j] = (if i == s then 1
else y[i]);
con3 {i in NODES}:  # одноразовий вихід з вершини i
sum{j in NODES: (j,i) in ARCS } x[j,i] = (if i == s then 1
else y[i]);
# цикл містить k вершин
con4: sum{i in NODES: i!=s} y[i] = k;
con5 {(i,j) in ARCS}:  z[i,j] - k*x[i,j] <= 0 ;
# обмеження (1.3.5)-(1.3.7) відповідають за зв'язність
циклу
# у задачі (1.3.1)-(1.3.10)
```

```
con6_1: sum{j in NODES: (s,j) in ARCS } z[s,j] = k;
con6_2: sum{j in NODES: (j,s) in ARCS } z[j,s] = 0;
con7 {i in NODES: i!=s}:
          sum{j in NODES: (i,j) in ARCS } z[i,j]
        - sum{j in NODES: (j,i) in ARCS } z[j,i] = - y[i];
# зв'язність циклу у задачі (3.11)-(3.18)
con15{(i,j) in ARCS: i!=s and j!=s}:
           u[i] - u[j] + k*x[i,j] <= k-1;

problem problem1: dk_min,x,y,z,con2,con3,con4,
# задача (1.3.1)-(1.3.10)
                  con5, con6_1, con6_2, con7;
problem problem2: dk_min,x,y,u,con2,con3,con4,con15;
# задача (1.3.11)-(1.3.18)

printf "st70.tsp data:\n" ;  # дані для графа st70.tsp

data;

param n:= 70;
param s:= 1;
param k:= 10;

param: xcoord :=
 1 64   2 80   3 69   4 72   5 48   6 58   7 81   8 79   9
30  10 42
11  7  12 29  13 78  14 64  15 95  16 57  17 40  18 68  19
92  20 62
21 28  22 76  23 67  24 93  25  6  26 87  27 30  28 77  29
78  30 55
31 82  32 73  33 20  34 27  35 95  36 67  37 48  38 75  39
8  40 20
41 54  42 63  43 44  44 52  45 12  46 25  47 58  48  5  49
90  50 41
51 25  52 37  53 56  54 10  55 98  56 16  57 89  58 48  59
81  60 29
61 17  62  5  63 79  64  9  65 17  66 74  67 10  68 48  69
83  70 84;
param: ycoord :=
 1 96   2 39   3 23   4  42   5 67   6 43   7 34   8 17   9
23  10 67
```

```
11 76  12 51  13 92  14   8  15 57  16 91  17 35  18 40  19
34  20  1
21 43  22 73  23 88  24  54  25  8  26 18  27  9  28 13  29
94  30  3
31 88  32 28  33 55  34  43  35 86  36 99  37 83  38 81  39
19  40 18
41 38  42 36  43 33  44  18  45 13  46  5  47 85  48 67  49
9  50 76
51 76  52 64  53 63  54  55  55  7  56 74  57 60  58 82  59
76  60 60
61 22  62 45  63 70  64 100  65 82  66 67  67 68  68 19  69
86  70 94;

for{(i,j) in ARCS} { # обчислюємо довжини дуг
  let d[i,j] := round(sqrt((xcoord[i]-
xcoord[j])*(xcoord[i]-xcoord[j])
             +(ycoord[i]-ycoord[j])*(ycoord[i]-
ycoord[j])));
}

# display d;
display n,k,s;

# option solver gurobi;
printf "problem1:\n" ;
solve problem1;
display _solve_time;

display dk_min;
printf "Display x:\n" ;
for{i in NODES} {
  for{j in NODES: (i,j) in ARCS and x[i,j] >= 0.02} {
     printf "%d -> %d [%0.2f]\n", i, j, x[i,j];
  }
}

printf "problem2:\n" ;
solve problem2;
display _solve_time;

display dk_min;
```

```
printf "Display x:\n" ;
for{i in NODES} {
  for{j in NODES: (i,j) in ARCS and x[i,j] >= 0.02} {
     printf "%d -> %d [%0.2f]\n", i, j, x[i,j];
  }
}

printf "Display u:\n" ;
for{i in NODES}{
   if y[i] > 0 then printf "%d\t %d\n" ,i,u[i];
}
```

```
Протокол роботи програми Gurobi 9.1.1
************************************************************
**

   NEOS Server Version 6.0
   Job#     : 10733941
   Password : kdInwMVG
   Solver   : milp:Gurobi:AMPL
   Start    : 2021-07-17 03:19:38
   End      : 2021-07-17 03:19:43
   Host     : prod-sub-1.neos-server.org


************************************************************
**
You are using the solver gurobi_ampl.

Checking ampl.mod for gurobi_options...
Executing AMPL.
processing data.
processing commands.
Executing on prod-exec-6.neos-server.org
st70.tsp data:
n = 70
k = 10
s = 1

problem1:

Presolve eliminates 70 constraints and 70 variables.
Adjusted problem:
9660 variables:
       4899 binary variables
       4761 linear variables
4972 constraints, all linear; 28980 nonzeros
       211 equality constraints
       4761 inequality constraints
1 linear objective; 4830 nonzeros.

Gurobi 9.1.1: threads=4
```

```
Gurobi 9.1.1: optimal solution; objective 74
8076 simplex iterations
1 branch-and-cut nodes
plus 1006 simplex iterations for intbasis
_solve_time = 3.30517

dk_min = 74

Display x:
1 -> 23 [1.00]
13 -> 29 [1.00]
22 -> 63 [1.00]
23 -> 38 [1.00]
29 -> 36 [1.00]
31 -> 13 [1.00]
36 -> 1 [1.00]
38 -> 22 [1.00]
59 -> 69 [1.00]
63 -> 59 [1.00]
69 -> 31 [1.00]
problem2:

Presolve eliminates 0 constraints and 4832 variables.
Adjusted problem:
4968 variables:
       4899 binary variables
       69 integer variables
4833 constraints, all linear; 23943 nonzeros
       141 equality constraints
       4692 inequality constraints
1 linear objective; 4830 nonzeros.

Gurobi 9.1.1: threads=4
Gurobi 9.1.1: optimal solution; objective 74
58947 simplex iterations
5808 branch-and-cut nodes
_solve_time = 8.50752

dk_min = 74

Display x:
```

```
1 -> 23 [1.00]
13 -> 29 [1.00]
22 -> 63 [1.00]
23 -> 38 [1.00]
29 -> 36 [1.00]
31 -> 13 [1.00]
36 -> 1 [1.00]
38 -> 22 [1.00]
59 -> 69 [1.00]
63 -> 59 [1.00]
69 -> 31 [1.00]
Display u:
13      8
22      3
23      1
29      9
31      7
36      10
38      2
59      5
63      4
69      6
```

```
Протокол роботи програми CPLEX 20.1.0.0
************************************************************
**

   NEOS Server Version 6.0
   Job#     : 10733945
   Password : KTmXpltz
   Solver   : milp:CPLEX:AMPL
   Start    : 2021-07-17 03:22:35
   End      : 2021-07-17 03:22:43
   Host     : prod-sub-1.neos-server.org

************************************************************
**
You are using the solver cplexamp.

Checking ampl.mod for cplex_options...
Executing AMPL.
processing data.
processing commands.
Executing on prod-exec-6.neos-server.org
st70.tsp data:
n = 70
k = 10
s = 1

problem1:

Presolve eliminates 70 constraints and 70 variables.
Adjusted problem:
9660 variables:
       4899 binary variables
       4761 linear variables
4972 constraints, all linear; 28980 nonzeros
       211 equality constraints
       4761 inequality constraints
1 linear objective; 4830 nonzeros.

CPLEX 20.1.0.0: threads=4
CPLEX 20.1.0.0: optimal integer solution; objective 74
```

```
2397 MIP simplex iterations
0 branch-and-bound nodes
_solve_time = 5.91247

dk_min = 74

Display x:
1 -> 36 [1.00]
13 -> 31 [1.00]
22 -> 38 [1.00]
23 -> 1 [1.00]
29 -> 13 [1.00]
31 -> 69 [1.00]
36 -> 29 [1.00]
38 -> 23 [1.00]
59 -> 63 [1.00]
63 -> 22 [1.00]
69 -> 59 [1.00]
problem2:

Presolve eliminates 0 constraints and 4832 variables.
Adjusted problem:
4968 variables:
       4899 binary variables
       69 integer variables
4833 constraints, all linear; 23943 nonzeros
       141 equality constraints
       4692 inequality constraints
1 linear objective; 4830 nonzeros.

CPLEX 20.1.0.0: threads=4
CPLEX 20.1.0.0: optimal integer solution; objective 74
62937 MIP simplex iterations
5215 branch-and-bound nodes
_solve_time = 14.4143

dk_min = 74

Display x:
1 -> 36 [1.00]
13 -> 31 [1.00]
```

```
22 -> 38 [1.00]
23 -> 1 [1.00]
29 -> 13 [1.00]
31 -> 69 [1.00]
36 -> 29 [1.00]
38 -> 23 [1.00]
59 -> 63 [1.00]
63 -> 22 [1.00]
69 -> 59 [1.00]
Display u:
13      3
22      8
23      10
29      2
31      4
36      1
38      9
59      6
63      7
69      5
```

ДОДАТОК В. Ratfor Програма Mulstr1

```
#  Текст програми mulstr1 на 16.05.2023
#  Розміри задачі (кількість галузей і ресурсів)
define MaxNsect   38# Максимальна кількість галузей
define MaxKr      10# Максимум ресурсів (ресурсні
обмеження)
define MaxVarB  1482# Максимум двосторонніх змінних deltaA
и deltaq
                    #  тобто не менш ніж
MaxNsect*(MaxNsect+1).
define MaxVar   1520# Максимум усіх змінних (+ додаткові z)
                    #  тобто не менш ніж MaxvarB+MaxNsect
implicit real*8 (a-h,o-z), integer*4 (i-n)
real*8 x(MaxVar)
common /Size/Nsect,Kr,NVarB  # галузі, ресурси, двосторонні
змінні
common
/bounds/xl(MaxVarB),xu(MaxVarB),irow(MaxVarB),icol(MaxVarB)
common /Penalty/Pen0,Pen1,Pen2  # штрафи для груп обмежень
common /opt/frobj,frcons,fobj,fcons,iobj,nls
common
/DVars/da(MaxNsect,MaxNsect),dq(MaxNsect),dz(MaxNsect)#оста
ння

#ітерація
common
/result/dar(MaxNsect,MaxNsect),dqr(MaxNsect),zr(MaxNsect)#р
екордна

#ітерація

common /Vars/a(MaxNsect,MaxNsect),q(MaxNsect),z(MaxNsect)
common
/Vectors/alpha(MaxNsect),dl(MaxNsect),dd(MaxNsect),dh(MaxNs
ect)
common /Resurs/br(MaxKr,MaxNsect,MaxNsect),Brhs(MaxKr)
common /Inflation/beta
```

```
iprint=8; open (iprint, file= 'output1.res') # файл друку
iwtr=2;  open (iwtr, file= 'mulstr1.res')  # запис
результатів (код помилки)

# читаємо та перевіряємо дані задачі
irdr=1; open (irdr,file= 'mulstr1.dat')
read (irdr,*)Nsect
if (Nsect>MaxNsect) {
  write (iprint,'(a,a)')' *** Error(1): кількість галузей',
                        ' перевищує максимально допустиму,'
  i1=MaxNsect
  write (iprint,'(2(a,i5))')'     т.е. Nsect =',Nsect,'
MaxNsect =',i1
  ierr=1; write (iwtr,'(1x,I12)')ierr
  stop
}
write (iprint,*)' Усього галузей: ',Nsect
write (iprint,*)' Коефіцієнти матриці витрат та межі на їх
зміни'
ind=1;
do j=1,Nsect {  # упорядковані за зростанням рядків,
стовпців
   do i=1,Nsect {
       read
(irdr,*)irow(ind),icol(ind),a(i,j),xl(ind),xu(ind)
       write (iprint,'(a,2i5,3(2x,f9.4))')' i j a(i,j) low
up ',
             irow(ind),icol(ind),a(i,j),xl(ind),xu(ind)
       if (irow(ind)!=i | icol(ind)!=j) {
         write (iprint,*) ' *** Error(1a): порушено порядок
коефіцієнтів'
         write (iprint,*) ' тобто потрібен коефіцієнт: i
j',i,j
         write (iprint,*) '      а заданий:  irow
jcol',irow(ind),icol(ind)
         ierr=11; write (iwtr,'(1x,I12)')ierr
         stop
       }
       if (a(i,j)>1.d0 | a(i,j)<0.d0) {
         write (iprint,*) ' *** Error(1b): неправильний
коефіцієнт матриці A'
```

```
         write (iprint,*) ' тобто не задовольняє умову -- 0
<= a(i,j) <=1'
         ierr=12; write (iwtr,'(1x,I12)')ierr
         stop
       }
       if (xu(ind)-xl(ind)<=-1.d-10) {
       write (iprint,*) ' *** Error(1c): неправильні межі
на змінні'
       write (iprint,*) 'тобто верхня межа менша за нижню'
         ierr=13; write (iwtr,'(1x,I12)')ierr
         stop
       }
       if (a(i,j)+xu(ind)>1.d0+1.d-10 |
a(i,j)+xl(ind)<0.d0-1.d-10) {
         write (iprint,*) ' *** Error(1d): неправильно
скориговані межі'
         write (iprint,*) ' тобто порушується 0<= a(i,j) +
delta a(i,j) <=1'
         ierr=14; write (iwtr,'(1x,I12)')ierr
         stop
       }
       if (dabs(xl(ind))<1.d-10 & dabs(xu(ind))<1.d-10)
next
       ind=ind+1
   }
}
write (iprint,*)' Коефіцієнти вектора q та межі на їх
зміну'
do i=1,Nsect {
   read (irdr,*)irow(ind),q(i),xl(ind),xu(ind)
   icol(ind)=0;
   write (iprint,'(a,i5,3(2x,f9.4))')' i q(i) low up
',irow(ind),q(i),
                                      xl(ind),xu(ind)
   if (irow(ind)!=i) {
     write (iprint,*) ' *** Error(2a): порушено порядок
коефіцієнтів'
     write (iprint,*) ' тобто потрібен коефіцієнт: i ',i
     write (iprint,*) '      a заданий:  irow ',irow(ind)
     ierr=21; write (iwtr,'(1x,I12)')ierr
     stop
```

```
   }
   if (q(i)>1.d0 | q(i)<0.d0) {
     write (iprint,*) ' *** Error(2b): неправильний
коефіцієнт вектора q'
     write (iprint,*) ' тобто не задовольняє умову -- 0 <=
q(i) <=1'
     ierr=22; write (iwtr,'(1x,I12)')ierr
     stop
   }
   if (xu(ind)-xl(ind)<=-1.d-10) {
     write (iprint,*) ' *** Error(2c): неправильні межі на
змінні'
     write (iprint,*) ' тобто верхня межа менша за нижню'
     ierr=23; write (iwtr,'(1x,I12)')ierr
     stop
   }
   if (q(i)+xu(ind)>1.d0+1.d-10 | q(i)+xl(ind)<0.d0-1.d-10)
{
     write (iprint,*) ' *** Error(2d): неправильно
скориговані межі'
     write (iprint,*) ' тобто порушується 0<= q(i) + delta
q(i) <=1'
     ierr=24; write (iwtr,'(1x,I12)')ierr
     stop
   }
   if (dabs(xl(ind))<1.d-10 & dabs(xu(ind))<1.d-10) next
   icol(ind)=0;  ind=ind+1
}
NVarB=ind-1; NVar=NVarB+Nsect
if (NVar>MaxVar | NVarB>MaxVarB ) {
  write (iprint,'(a,a)')' *** Error(3): кількість змінних',
                        ' перевищує максимально допустиму,'
  i1=MaxVar
  write (iprint,'(2(a,i5))')'     т.е. NVar =',NVar,'
MaxVar =',i1
  i1=MaxVarB
  write (iprint,'(2(a,i5))')'     т.е. NVarB =',NVarB,'
MaxVarB =',i1
  ierr=3; write (iwtr,'(1x,I12)')ierr
  stop
}
```

```
write (iprint,*) ' Вектори alpha, h, l, d '
do i=1,Nsect{ # читаємо та друкуємо вектори alpha,h,l,d
   read (irdr,*)ind,alpha(i),dh(i),dl(i),dd(i)
   write (iprint,'(a,i5,4(1x,f9.4))')'i alpha(i) h(i) l(i)
d(i) ',

ind,alpha(i),dh(i),dl(i),dd(i)
   if (ind!=i) {
     write (iprint,*) ' *** Error(3a): порушено порядок
даних',
                      ' про вектори alpha, h, l, d'
     write (iprint,*) ' тобто потрібен рядок i ',i
     write (iprint,*) '      а заданий: irow ',ind
     ierr=31; write (iwtr,'(1x,I12)')ierr
     stop
   }
}
read (irdr,*)Kr
if (Kr>MaxKr) {
  write (iprint,'(a,a)')' *** Error(4): кількість ресурсних
обмежень',
                        ' перевищує максимально допустиму,'
  i1=MaxKr
  write (iprint,'(2(a,i5))')'     т.е. Kr =',Kr,' MaxKr
=',i1
  ierr=4; write (iwtr,'(1x,I12)')ierr
  stop
}
write (iprint,'(a,i5)')' Дані про ресурсні обмеження:
всього ресурсів ',Kr
if (Kr>0) {
  read (irdr,*) (Brhs(k),k=1,Kr)
  write (iprint,'(a)')' Наявні ресурси'
  write (iprint,'(10f9.4)') (Brhs(k),k=1,Kr)
  write (iprint,'(a)')' Матриці витрат ресурсів (i-рядки,
j-стовпці).'
  do j=1,Nsect{
     do i=1,Nsect{
        read (irdr,*)ind1,ind2,(br(k,i,j),k=1,Kr)
        write (iprint,'(a,2i5,10f9.4)')' i j
(br(k,i,j),k=1,Kr) ',
```

```
ind1,ind2,(br(k,i,j),k=1,Kr)
        if (ind1!=i | ind2!=j) {
          write (iprint,*) ' *** Error(4a): порушено
порядок коефіцієнтів'
          write (iprint,*) ' тобто потрібен коефіцієнт: i
j',i,j
          write (iprint,*) '      a заданий:  irow
jcol',ind1,ind2
          ierr=41; write (iwtr,'(1x,I12)')ierr
          stop
        }
     }
  }
}
read (irdr,*)beta
write (iprint,*)' beta=',beta
close(irdr)  # закінчили читання та перевірку даних про
задачу

# друкуємо табличні дані про задачу (в економній формі)
write (iprint,'(a))') '                    ***** Матриця
технологій *****'
do i=1,Nsect
   write (iprint,'(5x,38(1x,f6.4))') (a(i,j),j=1,Nsect)
write (iprint,'(a))') 'alpha(i)    h(i)  l(i)  d(i) '
do i=1,Nsect
   write (iprint,'(38(1x,f7.4))')alpha(i),dh(i),dl(i),dd(i)
for(k=1; k<=Kr; k=k+1) {
    write (iprint,'(a,i3))') '   Матриця витрат ресурсу k =
',k
    do i=1,Nsect
       write (iprint,'(5x,38(1x,f7.2))')
(br(k,i,j),j=1,Nsect)
}

write (iprint,*)' Всього змінних у задачі:'
write (iprint,'(6x,a,I12)')' двосторонніх:   NVarB  = ',
NVarB
write (iprint,'(6x,a,I12)')' загалом:           NVar   = ',
NVar
```

```
# читаем параметры задачи
irdr=1; open(irdr,file='problem.dat')
read (irdr,*) iobj
read (irdr,*) Pen0
read (irdr,*) Pen1
read (irdr,*) Pen2
read (irdr,*) Npoint
close (irdr)

# друкуємо параметри задачі
write (iprint,'(a)') '  Параметри задачі: '
write (iprint,'(6x,a,i12)')  ' iobj     = ', iobj
write (iprint,'(6x,a,f12.5)')' Penalty0 = ', Pen0
write (iprint,'(6x,a,f12.5)')' Penalty1 = ', Pen1
write (iprint,'(6x,a,f12.5)')' Penalty2 = ', Pen2
write (iprint,'(6x,a,i12)')' Npoint   = ', Npoint

# читаємо параметри r-алгоритму
irdr=1; open(irdr,file='ralg.dat')
read (irdr,*) alp
read (irdr,*) h0
read (irdr,*) nh
read (irdr,*) q1
read (irdr,*) q2
read (irdr,*) intp
read (irdr,*) maxitn
read (irdr,*) epsx
read (irdr,*) epsg
close (irdr)

# друкуємо параметри r-алгоритму
write (iprint,'(a)') '  Параметри r-алгоритму: '
write (iprint,'(6x,a,f12.5)')' alpha  = ', alp
write (iprint,'(6x,a,f12.5)')' h0     = ', h0
write (iprint,'(6x,a,i12)')  ' nh     = ', nh
write (iprint,'(6x,a,f12.5)')' q1     = ', q1
write (iprint,'(6x,a,f12.5)')' q2     = ', q2
write (iprint,'(6x,a,i12)')  ' intp   = ',intp
write (iprint,'(6x,a,i8)')   ' maxitn = ',maxitn
write (iprint,'(6x,a,g12.5)')' epsx   = ',epsx
```

```
write (iprint,'(6x,a,g12.5)')' epsg   = ',epsg

ierr=0; write (iwtr,'(1x,I12)')ierr # далі записуватимемо
Npoint результатів
do ipoint=1,Npoint{
   write (iprint,'(a)')' Стартова точка для проблемних
змінних'
   xlam=dfloat(ipoint)/(dfloat(Npoint)+1.d0); xlam1=1.d0-
xlam;
   do i=1,NVarB{ # встановити початкову точку для
двосторонніх змінних
      x(i)=xlam1*xl(i)+xlam*xu(i)
      write (iprint,'(3x,a,i4,3(5x,f12.3))')'i x(i) xl(i)
xu(i)',

i,x(i),xl(i),xu(i)
   }
   do i=NVarB+1,NVar  # встановити початкову точку для
змінних z
      x(i)=0.d0
# Зауваження: для необмежених додаткових змінних z
               # стартову точку можна вибрати і краще, але
для цього треба
               # розв'язувати систему рівнянь, .... а поки
що взято нулі ...
    # Переведення стартової точки в ралговську згідно
заміни змінних 'пила'
    # (тільки для двосторонніх змінних)
   do i=1,NVarB{
      d1=1.d0; d2=1.d0; ddd=xu(i)-xl(i)
      if (ddd>1.d-12){
         d1=3.d0-2.d0*(x(i)-xl(i))/ddd
         d2=2.d0*(x(i)-xl(i))/ddd-1.d0
      }
      x(i)=d2; if (d1>1.d0) x(i)=d1
   }
   frobj=-1.d+20;  # встановити значення рекорду цільової
функції
   call
ralgb5(NVar,x,alp,h0,nh,q1,q2,maxitn,epsx,epsg,intp,itn,ist
op)
```

```
   write (iprint,'(3x,a,2(3x,i3))')   ' Ralg: itn
istop',itn,istop
   write (iprint,'(3x,a,2(3x,g12.5))')'     : fobj
frobj',fobj,frobj
   do i=1,NVar
      write (iprint,1000)'i x(i)',i,x(i)
   1000    format(6x,a,i4,2(5x,g12.5))
   write (iprint,*) ' frobj frcons ',frobj,frcons
   write (iprint,*) ' Матриця da'
   do i=1,Nsect
      write (iprint,'(38(2x,f5.3))') (da(i,j),j=1,Nsect)
   write (iprint,*) ' вектор dq'
   write (iprint,'(38(2x,f5.3))') (dq(i),i=1,Nsect)
   write (iprint,*) ' Матриця dar'
   do i=1,Nsect
      write (iprint,'(38(2x,f5.3))') (dar(i,j),j=1,Nsect)
   write (iprint,*) ' вектор dqr'
   write (iprint,'(38(2x,f5.3))') (dqr(i),i=1,Nsect)

   # записати знайдений розв'язок для стартової точки
   write (iwtr,'(1x,1PE13.5)') fobj
   write (iwtr,'(1x,1PE13.5)') fcons
   write (iwtr,'(1x,1PE13.5)') frobj
   write (iwtr,'(1x,1PE13.5)') frcons
   write (iwtr,'(1x,I12)') itn
   write (iwtr,'(1x,I12)') istop
   write (iwtr,'(1x,I12)') nls
   for(j=1; j<=nsect; j=j+1) {
      write (iwtr,'(1x,1PE13.5)') (dar(i,j),i=1,nsect)
   }
   write (iwtr,'(1x,1PE13.5)') (dqr(i),i=1,nsect)
   write (iwtr,'(1x,1PE13.5)') (zr(i),i=1,nsect)
}
close (iwtr)
stop
end #main
subroutine
ralgb5(n,x,alp,h0,nh,q1,q2,maxitn,epsx,epsg,intp,itn,istop)
implicit real*8(a-h,o-z),integer*4(i-n)
common /opt/frobj,frcons,fobj,fcons,iobj,nls
```

```
real*8
x(n),g(MaxVar),b(MaxVar,MaxVar),g1(MaxVar),g2(MaxVar)
#   Установка початкових параметрів
iprint=8
w=1./alp-1.
hs=h0; itn=0; lp=itn+intp
do i=1,n {
   do j=1,n
      b(j,i)=0.
   b(i,i)=1.
}
call calcfg(n,f,g,x)
dg=0.d0
do i=1,n
   dg=dg+g(i)*g(i)
istop=2
if (dsqrt(dg)<=epsg) goto 100
nls=0
nlsa=0
if (intp>=0) {
   write (iprint,3000)
   write (*,3000)
   write (iprint,3100) itn,f,frobj,nlsa,nls
   write (*,3100) itn,f,frobj,nlsa,nls
}
#  Основне тіло програми, що реалізує алгоритм
do itn=1,maxitn {
   # Обчислити субградієнт у перетвореному просторі
   dg2=0.d0
   do i=1,n{
      d=0.d0
      do j=1,n
         d=d+b(j,i)*g(j)
      g2(i)=d
      dg2=dg2+d*d
   }
   dg2=1.d0/dsqrt(dg2)
   do i=1,n
      g2(i)=dg2*g2(i)
   # Обчислити напрямок руху
   dg1=0.d0
```

```
do i=1,n {
   d=0.d0
   do j=1,n
      d=d+b(i,j)*g2(j)
   g1(i)=d
   dg1=dg1+d*d
}
dg1=dsqrt(dg1)
# Одновимірний спуск у напрямку
ls=0; ls1=0; lsa=0; dx=0.d0
20 continue
   ls=ls+1; ls1=ls1+1; dx=dx+hs*dg1
   do i=1,n
      x(i)=x(i)+hs*g1(i)
   call calcfg(n,f,g2,x)
   # Зупинка за нормою градієнта
   dg2=0.d0
   do i=1,n
      dg2=dg2+g2(i)*g2(i)
   if (ls1>nh) {
     hs=hs*q2
     ls1=0
   }
   istop=2
   if (dsqrt(dg2)<=epsg) goto 100
   istop=5
   if (ls>500) go to 100
   d=0.d0
   do i=1,n
      d=d+g1(i)*g2(i)
   if (d>0.d0) go to 20
if (ls==1) hs=hs*q1
nls=nls+ls
nlsa=nlsa+ls
if (itn==lp) {
  write (iprint,3100) itn,f,frobj,nlsa,nls
  write (*,3100) itn,f,frobj,nlsa,nls
  nlsa=0
  lp=lp+intp
}
istop=3
```

```
   if (dabs(dx)<epsx) goto 100
   # обчислити нормовану різницю субградієнтів
   do i=1,n
      g1(i)=g2(i)-g(i)
   dg=0.d0
   do i=1,n{
      d=0.d0
      do j=1,n
         d=d+b(j,i)*g1(j)
      g(i)=d
      dg=dg+d*d
   }
   dg=1.d0/dsqrt(dg)
   do i=1,n
      g(i)=dg*g(i)
   # Перерахувати матрицю B
   do i=1,n{
      d=0.d0
      do j=1,n
         d=d+b(i,j)*g(j)
      d=w*d
      do j=1,n
         b(i,j)=b(i,j)+d*g(j)
   }
   do i=1,n
      g(i)=g2(i)
}
itn=itn-1; istop=4
100 continue
if (intp>0) {
  write (iprint,3100) itn,f,frobj,nlsa,nls
  write (*,3100) itn,f,frobj,nlsa,nls
}
return
3000 format(/10x,' Протокол процесу негладкої оптимізації
'/2x,
              '  Itn.',7x,'..f(x)..',13x,'...f(x_r)...',
              5x,'LS',4x,'LSa')
3100 format(2x,i5,2(2x,1pd18.10),3(2x,i5))
end
```

```
subroutine calcfg(n,f,gf,x)  # обчислюємо значення функції
та суперградієнт
implicit real*8(a-h,o-z),integer*4(i-n)
common /Vars/a(MaxNsect,MaxNsect),q(MaxNsect),z(MaxNsect)
common
/DVars/da(MaxNsect,MaxNsect),dq(MaxNsect),dz(MaxNsect)
common
/Vectors/alpha(MaxNsect),dl(MaxNsect),dd(MaxNsect),dh(MaxNs
ect)
common /Resurs/br(MaxKr,MaxNsect,MaxNsect),Brhs(MaxKr)
common /Inflation/beta
common /Size/Nsect,Kr,NVarB
common /Penalty/Pen0,Pen1,Pen2
common
/bounds/xl(MaxVarB),xu(MaxVarB),irow(MaxVarB),icol(MaxVarB)
common /opt/frobj,frcons,fobj,fcons,iobj,nls
common
/result/dar(MaxNsect,MaxNsect),dqr(MaxNsect),zr(MaxNsect)
common
/Gradients/ga(MaxNsect,MaxNsect),gq(MaxNsect),gz(MaxNsect)

dimension gf(n),x(n)

# Обчислимо проблемні змінні задачі і збережемо їх у da, dq
та z.
do i=1,Nsect{     # для початку обнулимо da, dq
   dq(i)=0.d0
   do j=1,Nsect
       da(i,j)=0.d0
   z(i)=x(NVarB+i)   # вільні змінні
}

do i=1,NVarB{   # отримати двосторонні змінні (згідно з
"пилою")
   d=x(i); v=d
   if (d<-1.d0) v=d+4.d0*int((3.d0-d)/4.d0)
   if (d>3.d0)  v=d-4.d0*int((d+1.d0)/4.d0)
   x(i)=v; d1=0.5d0*(xu(i)-xl(i)); d2=0.5d0*(xu(i)+xl(i))
   if (v>1.d0) {
      t1=d1*(2.d0-v)+d2
   }
```

```
   else{
      t1=d1*v+d2
   }
   if (icol(i)>0)   da(irow(i),icol(i))=t1
   if (icol(i)==0)  dq(irow(i))=t1
}

fobj=0.d0     # обнулити значення функції та її градієнт
do i=1,n
   gf(i)=0.d0

# обчислити градієнт цільової функції (пов'язаний лише зі
змінними z)
if (iobj==1) {              # f(z) = (z,h)/(1.d0-(z,alpha))
   temp1=0.d0; temp2=1.d0
   do i=1,Nsect{
      temp1=temp1+z(i)*dh(i)
      temp2=temp2-z(i)*alpha(i)
   }
   fobj=fobj+temp1/temp2
   do i=1,Nsect{
      gz(i)=(dh(i)*temp2+temp1*alpha(i))/temp2/temp2
   }
}
if (iobj==2) {              # f(z) = (z,alpha)
Мультиплікатор Кейнса
   do i=1,Nsect{
      fobj=fobj+z(i)*alpha(i); gz(i)=alpha(i)
   }
}
if (iobj==0) {               # перевірка системи на
сумісність
   do i=1,Nsect
      gz(i)=0.d0
}

f=fobj # встановимо f рівним значенню цільової функції
       # і займемося вкладом у градієнт від штрафів

do i=1,Nsect{ # обнулимо ga и gq, куди будемо накопичувати
компоненти градієнта
```

```
   gq(i)=0.d0
   do j=1,Nsect
       ga(i,j)=0.d0
}

# Внесок від квадратичних обмежень-рівностей (dA, dq и z)
f1=0.d0
do i=1,Nsect {
   temp=q(i)+dq(i)-z(i)
   do j=1,Nsect
      temp=temp+(a(j,i)+da(j,i))*z(j)
   f1=f1+dabs(temp)  # суму модулей
   f=f-pen0*dabs(temp) # внесок у функцію та підрахунок
градієнтів
   one=1.d0
   if (temp<0.d0) one=-1.d0
   gq(i) = gq(i) + Pen0*one
   gz(i) = gz(i) + Pen0*one
   do j=1,Nsect {
       ga(j,i) = ga(j,i) + Pen0*z(j)*one
       gz(j) = gz(j) - Pen0*(a(j,i)+da(j,i))*one
   }
}

# Друга група обмежень (за винятком інфляції)

f21=0.d0 # перша частина (знаменник дробу додатний)
do j=1,Nsect{
   temp = a(j,j)+da(j,j) + dl(j)*(q(j)+dq(j)) + dd(j) -
1.d0
   if (temp>0.d0) {
      f21=f21+temp
      f=f-Pen1*temp  # запам'ятати та накопичити градієнти
      ga(j,j)=ga(j,j)+Pen1
      gq(j)=gq(j)+Pen1*dl(j)
   }
}

f22=0.d0 # друга частина (власне інфляція, помножена на
знаменник)
do j=1,Nsect{
```

```
   temp=0.d0
   do i=1,Nsect{
      if (i==j) next
      temp=temp+(a(i,j)+da(i,j))
   }
   temp=temp-beta*(1.d0-(a(j,j)+da(j,j))-
dl(j)*(q(j)+dq(j))-dd(j))
   if (temp<=0.d0) next
   f22=f22+temp;
   f=f - Pen1*temp  # запам'ятати та накопичити градієнти
   do i=1,Nsect{
      if (i==j) next
      ga(i,j)=ga(i,j) + Pen1
   }
   ga(j,j)=ga(j,j)+Pen1*beta
   gq(j)=gq(j)+Pen1*dl(j)*beta
}

f3=0.d0  # третя група обмежень (ресурсні обмеження)
for(k=1; k<=Kr; k=k+1){
   temp=-Brhs(k)
   do i=1,Nsect
      do j=1,Nsect{
          temp1=dabs(da(i,j))
          if (da(i,j)<0.d0) temp=temp+temp1*br(k,i,j)
       }
   if (temp<=0.d0) next
   f3=f3+temp
   f=f - Pen2*temp # запам'ятати та накопичити градієнти
   do i=1,Nsect
      do j=1,Nsect
         if (da(i,j)<0.d0) ga(i,j)=ga(i,j)-Pen2*br(k,i,j)
}

# корекція градієнта з урахуванням штрафу за da,dq
do i=1,NVarB{
   if (icol(i)>0)   gf(i)=gf(i)-ga(irow(i),icol(i))
   if (icol(i)==0)  gf(i)=gf(i)-gq(irow(i))
}
do i=1,Nsect
   gf(NvarB+i)=gf(NvarB+i)+gz(i)
```

```
# корекція градієнта з урахуванням заміни двосторонніх
змінних ('пила')
do i=1,NVarB{
   d1=0.5d0*(xu(i)-xl(i))
   gf(i)=gf(i)*d1
   if (x(i)>1.d0) gf(i)=-gf(i)
}

fobj=f; fcons=f1+f21+f22+f3
if (frobj<fobj) { # запам'ятаємо рекорні значення та точку
dqr(*),dar(*),zr(*)
  frobj=fobj
  do i=1,Nsect{
     do j=1,Nsect
        dar(i,j)=da(i,j)
     dqr(i)=dq(i); zr(i)=z(i)
  }
  frcons=fcons
}

return
end #calcfg


#############################################################
#Вхідні файли для тестового прикладу (див. Розділ 2)
#############################################################
#############################################################
# mulstr1.dat
#############################################################
7
1      1      0.337   -0.1685     0.1685
2      1      0.023   -0.0115     0.0115
3      1      0.163   -0.0815     0.0815
4      1      0.012   -0.006      0.006
5      1      0.009   -0.0045     0.0045
6      1      0.153   -0.0765     0.0765
7      1      0.161   -0.0805     0.0805
1      2      0.139   -0.0695     0.0695
2      2      0.251   -0.1255     0.1255
```

```
3       2       0.176   -0.088      0.088
4       2       0.009   -0.0045     0.0045
5       2       0.01    -0.005      0.005
6       2       0.121   -0.0605     0.0605
7       2       0.193   -0.0965     0.0965
1       3       0.215   -0.1075     0.1075
2       3       0.179   -0.0895     0.0895
3       3       0.191   -0.0955     0.0955
4       3       0.157   -0.0785     0.0785
5       3       0.008   -0.004      0.004
6       3       0.099   -0.0495     0.0495
7       3       0.103   -0.0515     0.0515
1       4       0.127   -0.0635     0.0635
2       4       0.089   -0.0445     0.0445
3       4       0.097   -0.0485     0.0485
4       4       0.031   -0.0155     0.0155
5       4       0.226   -0.113      0.113
6       4       0.031   -0.0155     0.0155
7       4       0.101   -0.0505     0.0505
1       5       0.146   -0.073      0.073
2       5       0.019   -0.0095     0.0095
3       5       0.103   -0.0515     0.0515
4       5       0.029   -0.0145     0.0145
5       5       0.107   -0.0535     0.0535
6       5       0.025   -0.0125     0.0125
7       5       0.095   -0.0475     0.0475
1       6       0.112   -0.056      0.056
2       6       0.131   -0.0655     0.0655
3       6       0.095   -0.0475     0.0475
4       6       0.026   -0.013      0.013
5       6       0.006   -0.003      0.003
6       6       0.019   -0.0095     0.0095
7       6       0.087   -0.0435     0.0435
1       7       0.196   -0.098      0.098
2       7       0.005   -0.0025     0.0025
3       7       0.087   -0.0435     0.0435
4       7       0.094   -0.047      0.047
5       7       0.0071  -0.0036     0.0036
6       7       0.033   -0.0165     0.0165
7       7       0.091   -0.0455     0.0455
1       0.05    0       0.9
```

```
2       0.02    0       0.9
3       0.01    0       0.9
4       0.08    0       0.9
5       0.09    0       0.9
6       0.12    0       0.88
7       0.14    0       0.86
1       0.15    0.1     1.32        0.01
2       0.05    0.2     1           0.05
3       0.05    0.2     1.375       0.01
4       0.15    0.1     1.375       0.05
5       0.2     0.05    1.375       0.1
6       0.2     0.05    1.375       0.1
7       0.2     0.3     1.375       0.15
1
3.5
1       1       3
2       1       3
3       1       3
4       1       3
5       1       3
6       1       3
7       1       3
1       2       3
2       2       3
3       2       3
4       2       3
5       2       3
6       2       3
7       2       3
1       3       3
2       3       3
3       3       3
4       3       3
5       3       3
6       3       3
7       3       3
1       4       1
2       4       1
3       4       1
4       4       1
5       4       1
```

```
6       4       1
7       4       1
1       5       1
2       5       1
3       5       1
4       5       1
5       5       1
6       5       1
7       5       1
1       6       2
2       6       2
3       6       2
4       6       2
5       6       2
6       6       2
7       6       2
1       7       0.5
2       7       0.5
3       7       0.5
4       7       0.5
5       7       0.5
6       7       0.5
7       7       0.5
0.95

#############################################################
# problem.dat
#############################################################
0
1
1
1
10

#############################################################
# ralg.dat
#############################################################
1.5
1
3
1
```

```
1.1
500
20000
0.0000001
0.0000001
############################################################
```

ДОДАТОК Г. Ratfor Програма Mulstr2

```
#  Текст програми  mulstr2 на 16.05.2023
#  Розмір задачі (кількість галузей)
define MaxNsect 38  # Максимальна кількість галузей
define MaxVarB  128 # Максимум двосторонніх змінних p і q
                    #  тобто не менш ніж MaxNsect+MaxNsect.
define MaxVar   512 # Максимум усіх змінних (+ додаткові z)
                    #  тобто не менш ніж MaxvarB+MaxNsect
implicit real*8 (a-h,o-z),integer*4 (i-n)
real*8 x(MaxVar)
common /Size/Nsect,NVarB        # галузі, двосторонні
змінні
common /bounds/xl(MaxVarB),xu(MaxVarB)
common /Penalty/Pen0,Pen1,Pen2  # штрафи для груп обмежень
common /opt/frobj,frcons,fobj,fcons,iobj,nls
common /DVars/p(MaxNsect),q(MaxNsect),z(MaxNsect)
#остання ітерація
common /result/pr(MaxNsect),qr(MaxNsect),zr(MaxNsect)
#рекордна ітерація

common /Vars/a(MaxNsect,MaxNsect)
common
/Vectors/alpha(MaxNsect),dl(MaxNsect),dd(MaxNsect),dh(MaxNs
ect)
common /Inflation/beta(MaxNsect)

iprint=8; open (iprint, file= 'output2.res') # файл друку
iwtr=2;  open (iwtr, file= 'mulstr2.res')    # запис
результатів (код помилки)

# читаємо та перевіряємо дані задачі
irdr=1; open (irdr,file='mulstr2.dat')
read (irdr,*)Nsect
if (Nsect>MaxNsect) {
  write (iprint,'(a,a)')' *** Error(1): кількість галузей',
                        ' перевищує максимально допустиму,'
  i1=MaxNsect
```

```
  write (iprint,'(2(a,i5))')'     тобто Nsect =',Nsect,'
MaxNsect =',i1
  ierr=1; write (iwtr,'(1x,I12)')ierr
  stop
}
write (iprint,*)' Всього галузей:: ',Nsect
write (iprint,*)' Коефіцієнти матриці витрат та межі на їх
зміни'
do j=1,Nsect {  # упорядковані за зростанням рядків,
стовпців
   do i=1,Nsect {
       read (irdr,*)k1,k2,a(i,j)
       write (iprint,'(a,2i5,3(2x,f9.4))')' i j a(i,j) ',
k1,k2,a(i,j)
       if (k1!=i | k2!=j) {
         write (iprint,*) ' *** Error(1a): порушено порядок
коефіцієнтів'
         write (iprint,*) ' тобто потрібен коефіцієнт: i
j',i,j
         write (iprint,*) '      а задано:  irow
icol',k1,k2
         ierr=11; write (iwtr,'(1x,I12)')ierr
         stop
       }
       if (a(i,j)>1.d0 | a(i,j)<0.d0) {
         write (iprint,*) ' *** Error(1b): неправильний
коефіцієнт матриці A'
         write (iprint,*) ' тобто не задовольняє умові -- 0
<= a(i,j) <=1'
         ierr=12; write (iwtr,'(1x,I12)')ierr
         stop
       }
   }
}
ind=1;
write (iprint,*)' Межі вектора p'
do i=1,Nsect {
   read (irdr,*)itemp1,xl(ind),xu(ind)
   write (iprint,'(a,i5,3(2x,f9.4))')' i low up ',itemp1,
               xl(ind),xu(ind)
   if (itemp1!=i) {
```

```
     write (iprint,*) ' *** Error(2a): порушено порядок
коефіцієнтів'
     write (iprint,*) ' тобто потрібен коефіцієнт: i ',i
     write (iprint,*) '      a задано:  irow ',itemp1
     ierr=21; write (iwtr,'(1x,I12)')ierr
     stop
   }
   if (xl(ind) < 0.d0 ) {
     write (iprint,*) ' *** Error(2b): невірна межа ',
                    ' вектора p (менше нуля)'
     ierr=22; write (iwtr,'(1x,I12)')ierr
     stop
   }
   if (xu(ind)-xl(ind)<=-1.d-10) {
     write (iprint,*) ' *** Error(2c): неправильні межі на
змінні p'
     write (iprint,*) ' тобто верхня межа менша за нижню'
     ierr=23; write (iwtr,'(1x,I12)')ierr
     stop
   }
   ind=ind+1
}
write (iprint,*)' Межі вектора q'
do i=1,Nsect {
   read (irdr,*)itemp1,xl(ind),xu(ind)
   write (iprint,'(a,i5,3(2x,f9.4))')' i low up ',itemp1,
                                     xl(ind),xu(ind)
   if (itemp1!=i) {
     write (iprint,*) ' *** Error(3a): порушено порядок
коефіцієнтів'
     write (iprint,*) ' тобто потрібен коефіцієнт: i ',i
     write (iprint,*) '      a задано:  irow ',itemp1
     ierr=31; write (iwtr,'(1x,I12)')ierr
     stop
   }
   if (xu(ind)>1.d0 | xl(ind)<0.d0) {
     write (iprint,*) ' *** Error(3b): невірна межа вектора
q'
     write (iprint,*) ' тобто не задовольняє умові -- 0 <=
q(i) <=1'
     ierr=32; write (iwtr,'(1x,I12)')ierr
```

```
     stop
   }
   if (xu(ind)-xl(ind)<=-1.d-10) {
     write (iprint,*) ' *** Error(3c): неправильні межі на
змінні'
     write (iprint,*) ' тобто верхня межа менша за нижню'
     ierr=33; write (iwtr,'(1x,I12)')ierr
     stop
   }
  ind=ind+1
}
NVarB=ind-1; NVar=NVarB+Nsect
if (NVar>MaxVar | NVarB>MaxVarB ) {
  write (iprint,'(a,a)')' *** Error(3): кількість змінних',
                        ' перевищує максимально допустиму,'
  i1=MaxVar
  write (iprint,'(2(a,i5))')'     тобто NVar =',NVar,'
MaxVar =',i1
  i1=MaxVarB
  write (iprint,'(2(a,i5))')'     тобто NVarB =',NVarB,'
MaxVarB =',i1
  ierr=3; write (iwtr,'(1x,I12)')ierr
  stop
}
write (iprint,*) ' Вектори alpha, h, l, d '
do i=1,Nsect{ # читаємо та друкуємо вектори alpha,h,l,d
   read (irdr,*)ind,alpha(i),dh(i),dl(i),dd(i)
   write (iprint,'(a,i5,4(1x,f9.4))')'i alpha(i) h(i) l(i)
d(i) ',

ind,alpha(i),dh(i),dl(i),dd(i)
   if (ind!=i) {
     write (iprint,*) ' *** Error(4a): порушений порядок у
даних',
                      ' про вектори alpha, h, l, d'
     write (iprint,*) ' тобто потрібен рядок i ',i
     write (iprint,*) '      а задано: irow ',ind
     ierr=41; write (iwtr,'(1x,I12)')ierr
     stop
   }
}
```

```
do i=1,NSect { # читаємо та друкуємо вектор beta
    read (irdr,*)ind, beta(i)
    write (iprint,'(a,i5,4(1x,f9.4))')'i beta(i) ',ind,
beta(i)
    if (ind!=i) {
        write (iprint,*) ' *** Error(5a): порушений порядок
у даних',
                         ' про вектор beta'
        write (iprint,*) ' тобто потрібен рядок i ',i
        write (iprint,*) '      а задано: irow ',ind
        ierr=51; write (iwtr,'(1x,I12)')ierr
        stop
    }
}
close(irdr)  # закінчили читання та перевірку даних про
задачу

# друкуємо табличні дані про задачу (в економній формі)
write (iprint,'(a))') '                    ***** Матриця
технологій *****'
do i=1,Nsect
   write (iprint,'(5x,38(1x,f6.4))') (a(i,j),j=1,Nsect)
write (iprint,'(a))') 'alpha(i)    h(i)  l(i)  d(i)
beta(i)'
do i=1,Nsect
   write
(iprint,'(38(1x,f7.4))')alpha(i),dh(i),dl(i),dd(i),beta(i)

write (iprint,*)' Разом змінних у задачі:'
write (iprint,'(6x,a,I12)')' двосторонніх:   NVarB  = ',
NVarB
write (iprint,'(6x,a,I12)')' всього:           NVar   = ',
NVar

# читаємо параметри задачі
irdr=1; open(irdr,file='problem.dat')
read (irdr,*) iobj
read (irdr,*) Pen0
read (irdr,*) Pen1
read (irdr,*) Pen2
read (irdr,*) Npoint
```

```
close (irdr)

# друкуємо параметри задачі
write (iprint,'(a)') '  Параметри Задачі: '
write (iprint,'(6x,a,i12)')  ' iobj     = ', iobj
write (iprint,'(6x,a,f12.5)')' Penalty0 = ', Pen0
write (iprint,'(6x,a,f12.5)')' Penalty1 = ', Pen1
write (iprint,'(6x,a,f12.5)')' Penalty2 = ', Pen2
write (iprint,'(6x,a,i12)')' Npoint   = ', Npoint

# читаємо параметри r-алгоритму
irdr=1; open(irdr,file='ralg.dat')
read (irdr,*) alp
read (irdr,*) h0
read (irdr,*) nh
read (irdr,*) q1
read (irdr,*) q2
read (irdr,*) intp
read (irdr,*) maxitn
read (irdr,*) epsx
read (irdr,*) epsg
close (irdr)

# друкуємо параметри r-алгоритму
write (iprint,'(a)') '  Параметри r-алгоритму: '
write (iprint,'(6x,a,f12.5)')' alpha  = ', alp
write (iprint,'(6x,a,f12.5)')' h0     = ', h0
write (iprint,'(6x,a,i12)')  ' nh     = ', nh
write (iprint,'(6x,a,f12.5)')' q1     = ', q1
write (iprint,'(6x,a,f12.5)')' q2     = ', q2
write (iprint,'(6x,a,i12)')  ' intp   = ',intp
write (iprint,'(6x,a,i8)')   ' maxitn = ',maxitn
write (iprint,'(6x,a,g12.5)')' epsx   = ',epsx
write (iprint,'(6x,a,g12.5)')' epsg   = ',epsg

ierr=0; write (iwtr,'(1x,I12)')ierr # далі записуватимемо
Npoint результатів
do ipoint=1,Npoint{
   write (iprint,'(a)')' Стартова точка щодо проблемних
змінних'
```

```
   xlam=dfloat(ipoint)/(dfloat(Npoint)+1.d0); xlam1=1.d0-
xlam;
   do i=1,NVarB{ # встановити початкову точку для
двосторонніх змінних
      x(i)=xlam1*xl(i)+xlam*xu(i)
      write (iprint,'(3x,a,i4,3(5x,f12.3))')'i x(i) xl(i)
xu(i)',

i,x(i),xl(i),xu(i)
   }
   do i=NVarB+1,NVar  # встановити початкову точку для
змінних z
      x(i)=0.d0
               # Зауваження: для необмежених додаткових
змінних z
                # стартову точку можна вибрати і краще, але
для цього треба
                # розв'язувати систему рівнянь, .... а поки
що взято нулі ...
    # Переведення стартової точки в ралговську згідно
заміни змінних 'пила'
    # (тільки для двосторонніх змінних)
   do i=1,NVarB{
      d1=1.d0; d2=1.d0; ddd=xu(i)-xl(i)
      if (ddd>1.d-12){
         d1=3.d0-2.d0*(x(i)-xl(i))/ddd
         d2=2.d0*(x(i)-xl(i))/ddd-1.d0
      }
      x(i)=d2; if (d1>1.d0) x(i)=d1
   }
   frobj=-1.d+20;  # встановити значення рекорду цільової
функції
   call
ralgb5(NVar,x,alp,h0,nh,q1,q2,maxitn,epsx,epsg,intp,itn,ist
op)
   write (iprint,'(3x,a,2(3x,i3))')   ' Ralg: itn
istop',itn,istop
   write (iprint,'(3x,a,2(3x,g12.5))')'     : fobj
frobj',fobj,frobj
   do i=1,NVar
      write (iprint,1000)'i x(i)',i,x(i)
```

```
   1000    format(6x,a,i4,2(5x,g12.5))
   write (iprint,*) ' frobj frcons ',frobj,frcons
   write (iprint,*) ' вектор p'
   write (iprint,'(38(2x,f5.3))') (p(i),i=1,Nsect)
   write (iprint,*) ' вектор q'
   write (iprint,'(38(2x,f5.3))') (q(i),i=1,Nsect)
   write (iprint,*) ' вектор pr'
   write (iprint,'(38(2x,f5.3))') (pr(i),i=1,Nsect)
   write (iprint,*) ' вектор qr'
   write (iprint,'(38(2x,f5.3))') (qr(i),i=1,Nsect)

   # записати знайдений розв'язок для стартової точки
   write (iwtr,'(1x,1PE13.5)') fobj
   write (iwtr,'(1x,1PE13.5)') fcons
   write (iwtr,'(1x,1PE13.5)') frobj
   write (iwtr,'(1x,1PE13.5)') frcons
   write (iwtr,'(1x,I12)') itn
   write (iwtr,'(1x,I12)') istop
   write (iwtr,'(1x,I12)') nls
   write (iwtr,'(1x,1PE13.5)') (pr(i),i=1,nsect)
   write (iwtr,'(1x,1PE13.5)') (qr(i),i=1,nsect)
   write (iwtr,'(1x,1PE13.5)') (zr(i),i=1,nsect)
}
close(iwtr)
stop
end #main
subroutine
ralgb5(n,x,alp,h0,nh,q1,q2,maxitn,epsx,epsg,intp,itn,istop)
implicit real*8(a-h,o-z),integer*4(i-n)
common /opt/frobj,frcons,fobj,fcons,iobj,nls
real*8
x(n),g(MaxVar),b(MaxVar,MaxVar),g1(MaxVar),g2(MaxVar)
#   Установка початкових параметрів
iprint=8
w=1./alp-1.
hs=h0; itn=0; lp=itn+intp
do i=1,n {
   do j=1,n
      b(j,i)=0.
   b(i,i)=1.
}
```

```
call calcfg(n,f,g,x)
dg=0.d0
do i=1,n
   dg=dg+g(i)*g(i)
istop=2
if (dsqrt(dg)<=epsg) goto 100
nls=0
nlsa=0
if (intp>=0) {
   write (iprint,3000)
   write (*,3000)
   write (iprint,3100) itn,f,frobj,nlsa,nls
   write (*,3100) itn,f,frobj,nlsa,nls
}
#  Основне тіло програми, що реалізує алгоритм
do itn=1,maxitn {
   # Обчислити субградієнт у перетвореному просторі
   dg2=0.d0
   do i=1,n{
      d=0.d0
      do j=1,n
         d=d+b(j,i)*g(j)
      g2(i)=d
      dg2=dg2+d*d
   }
   dg2=1.d0/dsqrt(dg2)
   do i=1,n
      g2(i)=dg2*g2(i)
   # Обчислити напрямок руху
   dg1=0.d0
   do i=1,n {
      d=0.d0
      do j=1,n
         d=d+b(i,j)*g2(j)
      g1(i)=d
      dg1=dg1+d*d
   }
   dg1=dsqrt(dg1)
   # Одновимірний спуск у напрямку
   ls=0; ls1=0; lsa=0; dx=0.d0
   20 continue
```

```
   ls=ls+1; ls1=ls1+1; dx=dx+hs*dg1
   do i=1,n
      x(i)=x(i)+hs*g1(i)
   call calcfg(n,f,g2,x)
   # Зупинка за нормою градієнта
   dg2=0.d0
   do i=1,n
      dg2=dg2+g2(i)*g2(i)
   if (ls1>nh) {
     hs=hs*q2
     ls1=0
   }
   istop=2
   if (dsqrt(dg2)<=epsg) goto 100
   istop=5
   if (ls>500) go to 100
   d=0.d0
   do i=1,n
      d=d+g1(i)*g2(i)
   if (d>0.d0) go to 20
if (ls==1) hs=hs*q1
nls=nls+ls
nlsa=nlsa+ls
if (itn==lp) {
  write (iprint,3100) itn,f,frobj,nlsa,nls
  write (*,3100) itn,f,frobj,nlsa,nls
  nlsa=0
  lp=lp+intp
}
istop=3
if (dabs(dx)<epsx) goto 100
# обчислити нормовану різницю субградієнтів
do i=1,n
   g1(i)=g2(i)-g(i)
dg=0.d0
do i=1,n{
   d=0.d0
   do j=1,n
      d=d+b(j,i)*g1(j)
   g(i)=d
   dg=dg+d*d
```

```
   }
   dg=1.d0/dsqrt(dg)
   do i=1,n
      g(i)=dg*g(i)
   # Перерахувати матрицю B
   do i=1,n{
      d=0.d0
      do j=1,n
         d=d+b(i,j)*g(j)
      d=w*d
      do j=1,n
         b(i,j)=b(i,j)+d*g(j)
   }
   do i=1,n
      g(i)=g2(i)
}
itn=itn-1; istop=4
100 continue
if (intp>0) {
  write (iprint,3100) itn,f,frobj,nlsa,nls
  write (*,3100) itn,f,frobj,nlsa,nls
}
return
3000 format(/10x,' Протокол процесу негладкої оптимізації
'/2x,
              '  Itn.',7x,'..f(x)..',13x,'...f(x_r)...',
              5x,'LS',4x,'LSa')
3100 format(2x,i5,2(2x,1pd18.10),3(2x,i5))
end
subroutine calcfg(n,f,gf,x)  # обчислюємо значення функції
та суперградієнт
implicit real*8(a-h,o-z),integer*4(i-n)
common /Vars/a(MaxNsect,MaxNsect)
common /DVars/p(MaxNsect),q(MaxNsect),z(MaxNsect)
#остання ітерація
common
/Vectors/alpha(MaxNsect),dl(MaxNsect),dd(MaxNsect),dh(MaxNs
ect)
common /Inflation/beta(MaxNsect)
common /Size/Nsect,NVarB
common /Penalty/Pen0,Pen1,Pen2
```

```
common /bounds/xl(MaxVarB),xu(MaxVarB)
common /opt/frobj,frcons,fobj,fcons,iobj,nls
common /result/pr(MaxNsect),qr(MaxNsect),zr(MaxNsect)
#рекордна ітерація
common /Gradients/gp(MaxNsect),gq(MaxNsect),gz(MaxNsect)

dimension gf(n),x(n)

# Обчислимо проблемні змінні задачі і збережемо їх у p, q
та z.
do i=1,Nsect{
   z(i)=x(NVarB+i)    # вільні змінні
}

do i=1,NVarB{    # отримати двосторонні змінні (згідно з
"пилою")
   d=x(i); v=d
   if (d<-1.d0) v=d+4.d0*int((3.d0-d)/4.d0)
   if (d>3.d0)  v=d-4.d0*int((d+1.d0)/4.d0)
   x(i)=v; d1=0.5d0*(xu(i)-xl(i)); d2=0.5d0*(xu(i)+xl(i))
   if (v>1.d0) {
      t1=d1*(2.d0-v)+d2
   }
   else{
      t1=d1*v+d2
   }

   if (i<=NSect) p(i) = t1
   if (i>NSect) q(i - NSect) = t1
}

fobj=0.d0      # обнулити значення функції та її градієнт
do i=1,n
   gf(i)=0.d0

# обчислити градієнт цільової функції (пов'язаний лише зі
змінними z)
if (iobj==1) {              # f(z) = (z,h)/(1.d0-(z,alpha))
   temp1=0.d0; temp2=1.d0
   do i=1,Nsect{
      temp1=temp1+z(i)*dh(i)
```

```
      temp2=temp2-z(i)*alpha(i)
   }
   fobj=fobj+temp1/temp2
   do i=1,Nsect{
      gz(i)=(dh(i)*temp2+temp1*alpha(i))/temp2/temp2
   }
}
if (iobj==2) {              # f(z) = (z,alpha)
Мультиплікатор Кейнса
   do i=1,Nsect{
      fobj=fobj+z(i)*alpha(i); gz(i)=alpha(i)
   }
}
if (iobj==0) {               # перевірка системи на
сумісність
   do i=1,Nsect
      gz(i)=0.d0
}

f=fobj # встановимо f рівним значенню цільової функції
       # і займемося вкладом у градієнт від штрафів

do i=1,Nsect{ # обнулимо ga и gq, куди будемо накопичувати
компоненти градієнта
    gp(i)=0.d0
    gq(i)=0.d0
}

# Обмеження 1"
# p_j q_j - p_j z_j + sum_i a_ij^0 p_i z_i = 0, j=1,n
f1=0.d0
do j=1,Nsect {
   temp=p(j)*(q(j) - z(j))
   do i=1,Nsect
      temp=temp+a(i,j)*p(i)*z(i)
   f1=f1+dabs(temp)     # суму модулей
   f=f-pen0*dabs(temp) # внесок у функцію та підрахунок
градієнтів
   Pen=Pen0
   if (temp<0.d0) Pen=-Pen0
   gp(j)    = gp(j) - Pen*(q(j)-z(j))
```

```
   gq(j)    = gq(j) - Pen*p(j)
   gz(j)    = gz(j) - Pen*(-p(j))
   do i=1,NSect{
    gp(i) = gp(i) - Pen*a(i,j)*z(i)
    gz(i) = gz(i) - Pen*a(i,j)*p(i)
   }
}

# Група обмежень 2-2'
# 2:  p_j - sum_i a_ij p_i - (l_j q_j + d_j) = 0, j=1,n
f21 = 0.d0
do j=1,NSect{
    temp=p(j)-(dl(j)*q(j)+dd(j))
    do i=1,NSect{
        temp = temp - a(i,j)*p(i)
    }
    f21 = f21 + dabs(temp)
    f = f - Pen1*dabs(temp)
    Pen=Pen1
    if (temp<0.d0) Pen=-Pen1
    gp(j) = gp(j) - Pen
    gq(j) = gq(j) - Pen*(-dl(j))
    do i=1,NSect{
        gp(i) = gp(i) - Pen*(-a(i,j))
    }
}
# 2": sum_i p_i = n
temp = -dfloat(NSect)
do i=1,NSect{
    temp = temp + p(i)
}
f22 = dabs(temp)
f = f - Pen1*dabs(temp)
Pen=Pen1
if (temp<0.d0) Pen=-Pen1
do i=1,NSect{
    gp(i) = gp(i) - Pen
}

# Обмеження 3
# sum_i a_ij p_i <= beta_j p_j, j=1,n
```

```
f3 = 0.d0
do j=1,NSect{
    temp = - beta(j)*p(j)
    do i=1,NSect{
        temp = temp + a(i,j) * p(i)
    }
    if (temp < 0.d0){
        next
    }
    f3 = f3 + dabs(temp)
    f = f - Pen2*temp
    gp(j) = gp(j) - Pen2*(- beta(j))
    do i=1,NSect{
        gp(i) = gp(i) - Pen2*a(i,j)
    }
}

# корекція градієнта з урахуванням штрафу за p,q
do i=1,NVarB{
    if (i <= NSect) gf(i)=gf(i)+gp(i)
    if (i > NSect) gf(i)=gf(i)+gq(i - NSect)
}
do i=1,Nsect
   gf(NvarB+i)=gf(NvarB+i)+gz(i)

# корекція градієнта з урахуванням заміни двосторонніх
змінних ('пила')
do i=1,NVarB{
   d1=0.5d0*(xu(i)-xl(i))
   gf(i)=gf(i)*d1
   if (x(i)>1.d0) gf(i)=-gf(i)
}

fobj=f; fcons=f1+f21+f22+f3
if (frobj<fobj) { # запам'ятаємо рекорні значення та точку
pr(*),qr(*),zr(*)
  frobj=fobj
  do i=1,Nsect{
     pr(i)=p(i); qr(i)=q(i); zr(i)=z(i)
  }
  frcons=fcons
```

```
}

return
end #calcfg


############################################################
# Вхідні файли для тестового прикладу (див. підрозділ 2.3)
############################################################
############################################################
# mulstr2.dat
############################################################
7
1       1           0.337
2       1           0.023
3       1           0.163
4       1           0.012
5       1           0.009
6       1           0.153
7       1           0.161
1       2           0.139
2       2           0.251
3       2           0.176
4       2           0.009
5       2           0.01
6       2           0.121
7       2           0.193
1       3           0.215
2       3           0.179
3       3           0.191
4       3           0.157
5       3           0.008
6       3           0.099
7       3           0.103
1       4           0.127
2       4           0.089
3       4           0.097
4       4           0.031
5       4           0.226
6       4           0.031
7       4           0.101
```

```
1       5           0.146
2       5           0.019
3       5           0.103
4       5           0.029
5       5           0.107
6       5           0.025
7       5           0.095
1       6           0.112
2       6           0.131
3       6           0.095
4       6           0.026
5       6           0.006
6       6           0.019
7       6           0.087
1       7           0.196
2       7           0.005
3       7           0.087
4       7           0.094
5       7           0.0071
6       7           0.033
7       7           0.091
1       0.00001     7
2       0.00001     7
3       0.00001     7
4       0.00001     7
5       0.00001     7
6       0.00001     7
7       0.00001     7
1       0.05        0.5
2       0.02        0.2
3       0.01        0.1
4       0.08        0.8
5       0.09        0.9
6       0.12        0.9
7       0.14        0.9
1       0.15        0.1     1.32    0.01
2       0.05        0.2     1       0.05
3       0.05        0.2     1.375   0.01
4       0.15        0.1     1.375   0.05
5       0.2         0.05    1.375   0.1
6       0.2         0.05    1.375   0.1
```

```
7      0.2        0.3    1.375  0.15
1      0.95
2      0.95
3      0.95
4      0.95
5      0.95
6      0.95
7      0.95

#############################################################
# problem.dat
#############################################################
0
1
1
1
10

#############################################################
# ralg.dat
#############################################################
1.5
1
3
1
1.1
500
20000
0.0000001
0.0000001
#############################################################
```

Наукове видання

П. І. Стецюк, М. Ю. Григорак, В. О. Стовба, О. А. Березовський,
Т. В. Бєлих, Є. А. Блохін, О. І. Воловик, А. А. Гонта, В. О. Жидков,
О. О. Жмуд, М. Г. Журбенко, О. П. Лиховид, В. В. Савицький,
В. П. Сизоненко, А. А. Супрун, О. М. Хом'як

МЕТОДИ НЕГЛАДКОЇ ОПТИМІЗАЦІЇ В ПРИКЛАДНИХ ЗАДАЧАХ

Відповідальний за макет:
доктор філософії Віктор Олександрович Стовба

В авторській редакції

Підписано до друку 20.07.2023. Зам. № 37-019
Гарнітура «Таймс». Друк офсетний. Папір офсетний.
Формат 60х84. Обл.-вид. арк. 7,25. Ум. друк. арк. 6,4.
Наклад 150 пр.

Віддруковано: ТОВ «ЛАЗУРИТ ПОЛІГРАФ»
04080, Київ, вул. Константинівська, 73
тел. (044) 463-73-01
email: lazurit_poligraf@ukr.net

Свідоцтво про внесення суб'єкта видавничої справи
до Державного реєстру видавців,
виготівників та розповсюджувачів видавничої продукції
серії ДК № 3133 від 17.03.2008