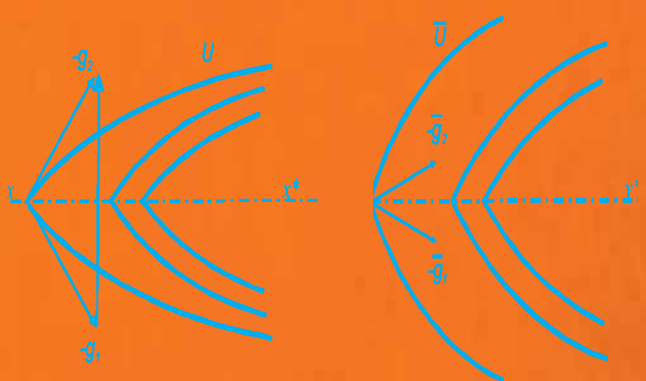
Субградієнтні алгоритми та задачі на комбінаторних конфігураціях

Субградієнтні алгоритми та задачі на комбінаторних конфігураціях



СУБГРАДІЄНТНІ АЛГОРИТМИ ТА ЗАДАЧІ НА КОМБІНАТОРНИХ КОНФІГУРАЦІЯХ

За загальною редакцією
доктора фізико-математичних наук П. І. Стецюка

Київ−2019
Університетське видавництво
ПУЛЬСАРИ

УДК 519.8
С 14

Автори:
П. І. Стецюк, Г. П. Донець, Е. І. Ненахов, Г. Ц. Чикрій,
О. А. Березовський, Т. В. Бєлих, В. І. Білецький, О. М. Хом'як,
О. О. Жмуд, А. В. Івлічев, О. І. Криворучко, В. О. Стовба

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.

Описані субградієнтні алгоритми з перетворенням простору та їхні реалізації мовою Octave. Ефективність алгоритмів підтверджена результатами тестування для задач мінімізації яружних опуклих функцій. Розглядаються методи розв'язання задачі про математичні сейфи, задачі розпізнавання предметів з нестандартними властивостями серед маси однотипних, оптимізаційних задач з лінійною, квадратичною та дробово-лінійною цільовими функціями на комбінаторних конфігураціях. Описано двоїстий підхід для розв'язання неопуклих квадратичних екстремальних задач та принцип розтягування часу під час ухвалення рішень в умовах конфлікту та невизначеності.

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

Рецензенти:
П. С. Кнопов, чл.-кор. НАН України
Л. Ф. Гуляницький, д-р техн. наук

Рекомендовано до друку Вченою радою
Інституту кібернетики імені В. М. Глушкова НАН України
(Протокол № 7 від 25 червня 2019 р.)

Публікація книги здійснена за фінансової підтримки НАН України
(проєкти № 0117U000327 і № 0118U005227)
та Volkswagen Foundation (грант № 90 306)

ISBN 978- 617-615-093-0

ВСТУП

У книзі представлено ряд результатів наукових досліджень відділу методів негладкої оптимізації та відділу економічної кібернетики Інституту кібернетики імені В. М. Глушкова НАН України, які пов'язані з виконанням науково-дослідних робіт за фінансової підтримки НАН України (проєкт № 0117U000327 «Розробити субградієнтні алгоритми розв'язання багатоекстремальних квадратичних оптимізаційних задач», 2017–2021 рр.; проєкт № 0117U000473 «Розробити методи та алгоритми розв'язання оптимізаційних задач на комбінаторних конфігураціях і траєкторіях динамічних систем», 2017–2021 рр.; проєкт № 0118U005227 «Розробити субградієнтні методи з перетворенням простору для спеціальних багатовимірних задач оптимізації», 2018–2019 рр.) та Volkswagen Foundation (грант № 90 306 «Оцінки помилок, критичні розв'язки, чисельні методи гладкої та негладкої оптимізації в задачах рівноваги, 2016–2019 рр.).

Розділ 1 присвячено трьом сімействам субградієнтних алгоритмів з перетворенням простору: у параграфі 1 розглядається модифікація r-алгоритму з постійним коефіцієнтом розтягування простору в напрямку різниці двох послідовних субградієнтів та адаптивним регулюванням кроку; у параграфі 2 розглядаються субградієнтні методи з кроком Поляка в перетвореному просторі змінних; у параграфі 3 розглядаються модифікації методу еліпсоїдів та їх застосування для розв'язання задачі мінімізації опуклої функції, задачі опуклого програмування та задачі пошуку сідлової точки опукло-увігнутої функції. У відповідних параграфах наведено опис реалізацій вказаних алгоритмів мовою Octave та підтверджено їх ефективність результатами тестування для задач мінімізації яружних квадратичних, кусково-лінійних та кусково-квадратичних функцій.

У розділі 2 розглядаються задачі комбінаторної оптимізації та методи їх розв'язання. У параграфі 4 представлені методи розв'язання задач про

математичні сейфи на графах та на (0-1)-матрицях. Розглянуто однотипові математичні сейфи з простим числом станів замків, математичні сейфи з довільною кількістю станів замків та математичні сейфи з різними типами замків. У параграфі 5 розглянуто задачі оптимізації лінійної та квадратичної функцій на перестановках, дробово-лінійної функції на комбінаторних конфігураціях, лінійної та квадратичної функції на комбінаторних конфігураціях з обмеженнями. У параграфі 6 описано методи комбінаторного розпізнавання активних куль серед маси подібних (розглядаються задачі двох типів: обмеженого та необмеженого комбінаторного розпізнавання).

У розділі 3 представлені результати досліджень, пов'язаних з розв'язанням наступних спеціальних задач. Параграф 7 присвячено застосуванню двоїстого підходу для розв'язання неопуклих квадратичних екстремальних задач. Сформульовано умови, при виконанні яких значення глобального екстремуму квадратичної екстремальної задачі та її двоїстої оцінки збігаються; наведено приклади їх застосування до конкретних задач для визначення випадків, коли знаходження двоїстої оцінки дозволяє знайти розв'язок задачі. У параграфі 8 описано методи розв'язання складних задач зближення рухомих об'єктів та результати аналізу процесів із запізненням інформації про фазовий стан, представлено принцип розтягування часу для випадку, коли не виконана умова Понтрягіна.

Представлені в книзі методи можуть використовуватися при розв'язанні широкого кола спеціальних задач в області дослідження операцій, які виникають у разі необхідності прийняття оптимальних рішень в управлінні, плануванні, проектуванні, моделюванні та ін. Розроблені програмні засоби можуть використовуватися як для розв'язання конкретних прикладних задач, так і в навчальному процесі для підготовки студентів відповідних спеціальностей.

Розділ 1. СУБГРАДІЄНТНІ АЛГОРИТМИ З ПЕРЕТВОРЕННЯМ ПРОСТОРУ

1. ТЕОРІЯ ТА ПРОГРАМНІ РЕАЛІЗАЦІЇ r-АЛГОРИТМІВ ШОРА

П. І. Стецюк, Т. В. Бєлих, О. І. Криворучко

Анотація. Розглядаються три обчислювальні форми r-алгоритмів, які відрізняються обсягом обчислень на одній ітерації. Наведено результати про збіжність граничного варіанта r-алгоритма для опуклих гладких функцій і $r_\mu(\alpha)$-алгоритма для опуклих кусочно-гладких функцій. Обговорюються практичні аспекти $r(\alpha)$-алгоритмів з постійним коефіцієнтом розтягу простору α та адаптивним способом регулювання кроку в напрямку нормованого антисубградієнта в перетвореному просторі змінних. Описана дослідницька програма **ralgb5a**, призначена для мінімізації гладких та негладких опуклих функцій.

Annotation. Three computational forms of r-algorithms that differ in the number of computations at iteration are considered. The results of convergence of the limit version of r-algorithm for convex smooth functions and $r_\mu(\alpha)$-algorithm for convex piecewise-smooth functions are presented. The practical aspects of $r(\alpha)$-algorithms with constant coefficient of space dilation α, and adaptive adjustment of step coefficient in the direction of the normalized antisubgradient in the transformed space of variables are discussed. The research program **ralgb5a** to minimize smooth and non-smooth convex functions is described.

1.1. Вступ

Субградієнтні методи з розтягом простору в напрямку різниці двох послідовних субградієнтів отримали назву r-алгоритмів [1, 2, 3]. Програмні реалізації r-алгоритмів виявилися достатньо конкурентоспроможними як за

часом обчислень, так і за точністю результатів з найбільш ефективними методами розв'язання гладких погано обумовлених задач. Прискорену збіжність r-алгоритмів при мінімізації негладких опуклих функцій забезпечує тісний взаємозв'язок двох принципів у чисельних методах оптимізації.

Перший принцип полягає у використанні процедури найшвидшого спуску в напрямку антисубградієнта опуклої функції в перетвореному просторі змінних. Якщо пошук мінімуму функції здійснюється точно, то перший принцип гарантує монотонність значень опуклої функції для точок мінімізуючої послідовності, яка будується за допомогою r-алгоритмів. Якщо пошук мінімуму функції здійснюється наближено, то «монотонність» по мінімізуючій функції замінюється на «майже монотонність». Однак найшвидший спуск для негладких функцій може зациклитися, тому для запобігання цьому використовується другий метод.

Другий принцип спрямований на зменшення ступеня витягнутості поверхонь рівня яружних функцій у перетвореному просторі змінних. Він полягає у використанні операції розтягу простору в напрямку різниці двох послідовних субградієнтів, де другий субградієнт обчислений у точці мінімуму функції в напрямку першого антисубградієнта. В результаті цього розтягу зменшуються поперечні складові субградієнтів уздовж напрямку точки мінімуму, що забезпечує достатньо швидку збіжність субградієнтного процесу з розтягом простору.

Комбінації цих принципів за певного регулювання кроку найшвидшого спуску (точного чи наближеного) і відповідного вибору коефіцієнта розтягу простору забезпечують прискорену збіжність конкретних варіантів r-алгоритмів і гарантують їхню монотонність (або майже монотонність) за значеннями мінімізуючої функції. Це підтверджується результатами численних застосувань r-алгоритмів у задачах лінійного і нелінійного програмування, блочних задачах з різними схемами декомпозиції, при вирішенні мінімаксних і матричних задач оптимізації, для обчислення двоїстих лагранжевих оцінок у багатоекстремальних і комбінаторних задачах оптимізації [4, 5, 6, 7].

1.2. Три обчислювальні форми r-алгоритмів

Розглядається задача мінімізації опуклої функції $f(x)$, де $x \in E^n$ — вектор із n змінних. Мінімальне значення функції позначатимемо $f^* = f(x^*)$, $x^* \in X^*$. Також припустимо, що $f(x)$ має обмежену множину мінімумів X^*, тобто виконується умова $\lim\limits_{\|x\|\to\infty} f(x) = +\infty$. Ця умова забезпечує коректність регулювання кроку в r-алгоритмах. Нехай $\{\alpha_k\}_{k=0}^{\infty}$ — набір коефіцієнтів розтягу простору, таких що $\alpha_k > 1$.

r-Алгоритмом для мінімізації $f(x)$ називається ітеративна процедура знаходження послідовності n-вимірних векторів $\{x_k\}_{k=0}^{\infty}$ і послідовності $n \times n$-матриць $\{B_k\}_{k=0}^{\infty}$ за таким правилом:

$$x_{k+1} = x_k - h_k B_k \xi_k, \quad B_{k+1} = B_k R_{\beta_k}(\eta_k), \quad k = 0,1,2,\ldots, \tag{1.1}$$

де

$$\xi_k = \frac{B_k^T g_f(x_k)}{\left\| B_k^T g_f(x_k) \right\|}, \quad h_k \geq h_k^* = \arg\min_{h \geq 0} f(x_k - h B_k \xi_k), \tag{1.2}$$

$$\beta_k = \frac{1}{\alpha_k} < 1, \quad \eta_k = \frac{B_k^T r_k}{\| B_k^T r_k \|}, \quad \text{де} \quad r_k = g_f(x_{k+1}) - g_f(x_k), \tag{1.3}$$

де x_0 — початкова точка; $B_0 = I_n$ — одинична $n \times n$-матриця; h_k^* — величина кроку за умови мінімуму функції $f(x)$ у напрямку нормованого антисубградієнта в перетвореному просторі змінних; $R_\beta(\eta) = I_n + (\beta - 1)\eta\eta^T$ — оператор стиснення простору субградієнтів у нормованому напрямку η з коефіцієнтом $\beta = \dfrac{1}{\alpha} < 1$; $g_f(x_k)$ і $g_f(x_{k+1})$ — субградієнти функції $f(x)$ у точках x_k і x_{k+1}.

Якщо на ітерації k для процесу (1.1)–(1.3) виконуються певні критерії (умови) зупинки, то покладемо $k^* = k$, $x_k^* = x_k$ і закінчимо роботу алгоритму.

На кожній ітерації r-алгоритмів реалізується субградієнтний спуск для опуклої функції $\varphi(y) = f(B_k y)$ у перетвореному просторі змінних $y = A_k x$, де $A_k = B_k^{-1}$.

Якщо ж обидві частини формули $x_{k+1} = x_k - h_k B_k \xi_k$ помножити зліва на матрицю A_k, то отримаємо

$$y_{k+1} = A_k x_{k+1} = A_k x_k - h_k \xi_k = y_k - h_k \frac{B_k^T g_f(x_k)}{\left\| B_k^T g_f(x_k) \right\|} = y_k - h_k \frac{g_\varphi(y_k)}{\left\| g_\varphi(y_k) \right\|}, \qquad (1.4)$$

де вектор $g_\varphi(y_k) = B_k^T g_f(x_k)$ є субградієнтом функції $\varphi(y) = f(B_k y)$ у точці $y_k = A_k x_k$ простору змінних $y = A_k x$. Це випливає з того, що субградієнт функції $f(x)$ у точці x_k задовольняє нерівність

$$f(x) \geq f(x_k) + (g_f(x_k))^T (x - x_k) \quad \forall x \in E^n,$$

звідки, виконавши заміну змінних $x = B_k y$, одержимо

$$\varphi(y) \geq \varphi(y_k) + (B_k^T g_f(x_k))^T (y - y_k) = \varphi(y_k) + (g_\varphi(y_k))^T (y - y_k) \quad \forall y \in E^n.$$

Якщо $h_k = h_k^*$, то формула (1.4) означає точний пошук мінімуму функції $\varphi(y) = f(B_k y)$ у напрямку нормованого антисубградієнта в перетвореному просторі змінних $y = A_k x$; якщо ж $h_k \approx h_k^*$, то (1.4) означає наближений пошук.

Якщо функція $f(x)$ є недиференційовною в точці x_k, то можливим є випадок $h_k = h_k^* = 0$, з яким пов'язані основні проблеми з умовою зупинки r-алгоритмів для негладких функцій. Якщо $h_k = 0$, то це не означає, що в точці x_k процес спуску потрібно зупинити.

У процесі виконання серії ітерацій з нульовим кроком наступна точка $x_{k+1} = x_k$ не змінюється, але змінюється матриця B_{k+1} і наступний субградієнт $g_f(x_{k+1})$. Це означає, що за рахунок послідовних розтягувань простору в напрямку різниці двох субградієнтів (обидва субградієнти отримані в точці x_k) здійснюється пошук потрібного напрямку спадання функції з точки x_k.

Монотонність по мінімізуючій функції для r-алгоритмів забезпечує саме розтяг простору в напрямку різниці двох послідовних субградієнтів. Саме цим r-алгоритми принципово відрізняються від субградієнтних методів з розтягом простору в напрямку субградієнта, які для негладких опуклих функцій взагалі не можуть бути монотонними по мінімізуючій функції. Покажемо це на прикладі кусково-гладкої функції (див. рис. 1.1).

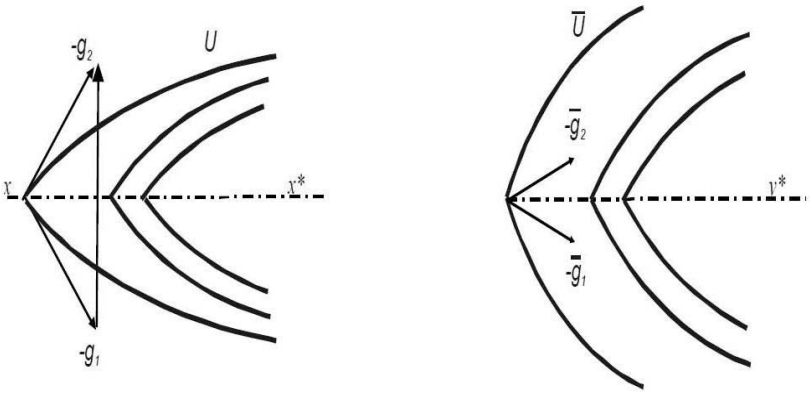


Рис. 1.1. Після розтягу простору за різницею двох субградієнтів (ліворуч) антисубградієнти стають напрямками спадання функції (праворуч)

Нехай ми знаходимося на межі двох «кусків» кусково-гладкої поверхні рівня, а градієнти до цих гладких «кусків», обчислені в цій точці, утворюють тупий кут (рисунок ліворуч). Жодний розтяг простору в напрямку градієнтів не може перетворити цей кут на гострий, а може лише наблизити його до $\pi/2$, залишаючи тупим. Тому, застосовуючи розтяг простору в напрямку субградієнта, неможливо отримати напрямок спадання функції у вигляді антиградієнта до одного з «кусків» у розтягнутому просторі.

Слід зауважити, що розтяг простору в напрямку різниці двох наведених градієнтів з достатнім коефіцієнтом розтягу перетворює тупий кут між

градієнтами на гострий, тобто відповідні образи цих антиградієнтів у розтягнутому просторі стають напрямками спадання функції (рисунок праворуч).

Отже, сімейство r-алгоритмів визначається послідовністю коефіцієнтів розтягу простору $\{\alpha_k\}_{k=0}^{\infty}$, послідовністю величин кроків $\{h_k\}_{k=0}^{\infty}$ та критеріями зупинки. При цьому величина кроку обирається з умови точного (наближеного) мінімуму функції у напрямку антисубградієнта в перетвореному просторі змінних, завдяки чому визначаються два послідовних субградієнти, розтяг по різниці яких покращує властивості яружної функції у перетвореному просторі змінних. За певного регулювання кроку та коефіцієнтів розтягу простору r-алгоритми є монотонними (чи майже монотонними) по мінімізуючій функції.

Метод (1.1)–(1.3) називають B-формою r-алгоритмів, на кожній його ітерації коригується матриця, пов'язана з заміною змінних $x = By$. Ітерація методу потребує порядку $5n^2$ арифметичних операцій множення, які визначають обчислювальну трудоємність ітерації (операціями додавання знехтуємо через їхній малий внесок у трудоємність ітерації). З них $3n^2$ множень необхідні для обчислення векторів $B_k\xi_k$, $B_k^T g_f(x_k)$ та $B_k^T r_k$ (множення матриці на вектор), а $2n^2$ множень необхідні для однорангової корекції матриці $B_{k+1} = B_k R_{\beta_k}(\eta_k)$. Справді,

$$B_{k+1} = B_k R_{\beta_k}(\eta_k) = B_k(I_n + (\beta_k - 1)\eta_k\eta_k^T) = B_k + (\beta_k - 1)(B_k\eta_k)\eta_k^T,$$

звідки легко побачити, що обчислення вектора $\eta = B_k\eta_k$ потребує стільки ж n^2 множень, скільки і побудова однорангової матриці $\eta\eta_k^T$.

У r-алгоритмів є ще одна B-форма, яка порівняно з методом (1.1)–(1.3) дозволяє заощадити n^2 операцій множення на кожній ітерації. Економні r-алгоритми описуються ітеративною процедурою знаходження послідовності векторів $\{x_k\}_{k=0}^{\infty}$ і матриць $\{B_k\}_{k=0}^{\infty}$ за таким правилом:

$$x_{k+1} = x_k - h_k B_k \frac{\tilde{g}_k}{\|\tilde{g}_k\|}, \quad B_{k+1} = B_k R_{\beta_k}(\eta_k), \quad k = 0,1,2,\ldots, \tag{1.5}$$

де

$$h_k \geq h_k^* = \arg\min_{h\geq 0} f(x_k - hB_k \frac{\tilde{g}_k}{\|\tilde{g}_k\|}), \quad g_{k+1}^* = B_k^T g_f(x_{k+1}), \tag{1.6}$$

$$\beta_k = \frac{1}{\alpha_k}, \quad \eta_k = \frac{g_{k+1}^* - \tilde{g}_k}{\| g_{k+1}^* - \tilde{g}_k \|}, \quad \tilde{g}_{k+1} = R_{\beta_k}(\eta_k) g_{k+1}^*, \tag{1.7}$$

де x_0 — початкова точка, така, що $x_0 \neq x^*$; $\tilde{g}_0 = B_0^T g_f(x_0)$, B_0 — невироджена $n \times n$-матриця; $g_f(x_k)$, $g_f(x_{k+1})$ — субградієнти функції $f(x)$ у точках x_k і x_{k+1}.

Якщо на ітерації k виконуються деякі критерії зупинки, то покладемо $k^* = k$, $x_k^* = x_k$ і закінчимо роботу алгоритму.

Ітерація методу (1.5)–(1.7) потребує $4n^2$ операцій множення. З них $2n^2$ множень потрібні для обчислення векторів $B_k\tilde{g}_k$ і $B_k^T g_f(x_{k+1})$, а $2n^2$ множень — для однорангової корекції матриці $B_{k+1} = B_k R_{\beta_k}(\eta_k)$. Економія n^2 множень пов'язана з тим, що $\tilde{g}_{k+1} = B_{k+1}^T g_f(x_{k+1})$, субградієнт у просторі змінних $y = A_{k+1}x$ перераховується з урахуванням уже обчисленого $g_{k+1}^* = B_k^T g_f(x_{k+1})$ — субградієнта в просторі змінних $y = A_k x$. Перерахунок субградієнта $\tilde{g}_{k+1}$ обчислюється за формулою

$$\tilde{g}_{k+1} = B_{k+1}^T g_f(x_{k+1}) = R_{\beta_k}(\eta_k) B_k^T g_f(x_{k+1}) = R_{\beta_k}(\eta_k) g_{k+1}^* =$$

$$= (I_n + (\beta_k - 1)\eta_k \eta_k^T) g_{k+1}^* = g_{k+1}^* + (\beta_k - 1)(\eta_k^T g_{k+1}^*)\eta_k,$$

яка не потребує операції множення матриці на вектор. При перерахунку виникає більше помилок при обчисленні нормованого субградієнта в перетвореному просторі за формулою $\xi_k = \frac{\tilde{g}_k}{\|\tilde{g}_k\|}$, ніж при обчисленні нормованого субградієнта $\xi_k = \frac{B_k^T g_f(x_k)}{\|B_k^T g_f(x_k)\|}$, як у методі (1.1)–(1.3).

r-Алгоритми можна записати в H-формі за допомогою симетричної матриці $H_k = B_k B_k^T$ (за аналогією з методами змінної метрики). Їм відповідає ітеративна процедура знаходження послідовності векторів $\{x_k\}_{k=0}^{\infty}$ і симетричних матриць $\{H_k\}_{k=0}^{\infty}$ за таким правилом:

$$x_{k+1} = x_k - h_k \frac{H_k g_f(x_k)}{\sqrt{\left(g_f(x_k)\right)^T H_k g_f(x_k)}},$$
$$H_{k+1} = H_k + (\beta_k^2 - 1)\frac{H_k r_k r_k^T H_k}{r_k^T H_k r_k}, \quad k = 0,1,\ldots, \tag{1.8}$$

де

$$h_k \geq h_k^* = \arg\min_{h\geq 0} f\left(x_k - h\frac{H_k g_f(x_k)}{\sqrt{\left(g_f(x_k)\right)^T H_k g_f(x_k)}}\right),$$
$$\beta_k = \frac{1}{\alpha_k} < 1, \quad r_k = g_f(x_{k+1}) - g_f(x_k), \tag{1.9}$$

де x_0 — початкова точка; $H_0 = I_n$ — одинична $n\times n$-матриця; h_k — величина кроку, яка не менша, ніж h_k^*; $g_f(x_k)$ і $g_f(x_{k+1})$ — субградієнти функції $f(x)$ у точках x_k і x_{k+1}. Якщо на ітерації k виконуються умови зупинки, то покладемо $k^* = k$, $x_k^* = x_k$ і завершуємо роботу алгоритму.

У H-формі r-алгоритмів формула для перерахунку наступного наближення x_{k+1} випливає із справедливості такого ланцюжка співвідношень

$$B_k \frac{B_k^T g_f(x_k)}{\left\|B_k^T g_f(x_k)\right\|} = \frac{B_k B_k^T g_f(x_k)}{\sqrt{\left(B_k^T g_f(x_k)\right)^T B_k g_f(x_k)}} =$$
$$= \frac{H_k g_f(x_k)}{\sqrt{\left(g_f(x_k)\right)^T B_k B_k^T g_f(x_k)}} = \frac{H_k g_f(x_k)}{\sqrt{\left(g_f(x_k)\right)^T H_k g_f(x_k)}}.$$

Формула для перерахунку симетричної матриці H_{k+1} випливає з такого ланцюжка перетворень

$$H_{k+1} = B_{k+1}B_{k+1}^T = B_k R_{\beta_k}(\eta_k)\left(B_k R_{\beta_k}(\eta_k)\right)^T = B_k R_{\beta_k}(\eta_k) R_{\beta_k}^T(\eta_k) B_k^T =$$

$$= B_k R_{\beta_k}(\eta_k) R_{\beta_k}(\eta_k) B_k^T = B_k R_{\beta_k^2}(\eta_k) B_k^T = B_k (I_n + (\beta_k^2 - 1)\eta_k \eta_k^T) B_k^T =$$

$$= B_k B_k^T + (\beta_k^2 - 1) B_k \eta_k \eta_k^T B_k^T = H_k + (\beta_k^2 - 1)\frac{B_k B_k^T r_k r_k^T B_k B_k^T}{\left\| B_k^T r_k \right\|^2} =$$

$$= H_k + (\beta_k^2 - 1)\frac{H_k r_k r_k^T H_k}{\left(B_k^T r_k\right)^T B_k^T r_k} = H_k + (\beta_k^2 - 1)\frac{H_k r_k r_k^T H_k}{r_k^T B_k B_k^T r_k} = H_k + (\beta_k^2 - 1)\frac{H_k r_k r_k^T H_k}{r_k^T H_k r_k}.$$

H-форма r-алгоритмів економніша, ніж B-форма. За оперативною пам'яттю — майже удічі, оскільки потрібно зберігати симетричну матрицю, а за трудоємністю — мінімум у 1,66 раза. Навіть якщо симетричну матрицю H_k зберігати як повну матрицю розмірністю $n \times n$, то ітерація методу (1.8)–(1.9) потребує $3n^2$ операцій множення. $2n^2$ множень необхідно для обчислення векторів $H_k g_f(x_k)$ і $\eta = H_k r_k$, а n^2 множень — для обчислення однорангової матриці $\eta\eta^T = H_k r_k r_k^T H_k$, яка використовується при перерахунку матриці H_{k+1}. Проте, H-форма r-алгоритмів обчислювально менш стійка, ніж B-форма.

Так, при реалізації методу (1.8)–(1.9) необхідно контролювати, щоб матриця H_k була додатньо визначеною. У методах (1.1)–(1.3) і (1.5)–(1.7) такий контроль не потрібен, оскільки обчислення пов'язані з додатньо визначеною матрицею $H_k = B_k B_k^T$.

Обчислювальні характеристики описаних форм r-алгоритмів за пам'яттю і трудоємністю наведені в табл. 1.1. Хоча теоретично всі три форми r-алгоритмів однакові, обчислювальна стійкість їхніх комп'ютерних реалізацій, що подана в останньому стовпчику таблиці, відрізняється.

Для обох B-форм вона позначена як «хороша», але перевагу віддано методу (1.1)–(1.3) і відзначено знаком «+». Це обумовлено тим, що перерахунок субградієнта при переході в наступний простір змінних, що використовується

в економній B-формі, сприяє збільшенню помилок щодо обчислення цього ж субградієнта в методі (1.1)–(1.3).

Обчислювальна стійкість r-алгоритмів в H-формі відзначена як «середня». Це означає, що за допомогою них взагалі неможливо знайти такі наближення до точки мінімуму, які можна знайти з r-алгоритмів у B-формі. r-Алгоритми в H-формі можна використовувати, якщо не вимагається високої точності для знаходження мінімуму функції $f(x)$.

Табл.. 1.1. Обчислювальні характеристики трьох форм r-алгоритмів

Форма r-алгоритмів	Вид методу	Оперативна пам'ять	Трудомісткість ітерації	Стійкість методу
B-форма	метод (1.1)–(1.3)	$\sim n^2$	$\sim 5n^2$	хороша (+)
економна B-форма	метод (1.5)–(1.7)	$\sim n^2$	$\sim 4n^2$	хороша
H-форма	метод (1.8)–(1.9)	$\sim n^2/2$	$\sim 3n^2$	середня

1.3. Три теореми про збіжність r-алгоритмів

Для опуклих функцій теоретичні результати про збіжність r-алгоритмів пов'язані з їхніми модифікаціями: з граничним варіантом r-алгоритмів для гладких функцій [1] та $r_\mu(\alpha)$-алгоритмом для негладких функцій [8].

Перший результат полягає в тому, що граничний варіант r-алгоритмів є проективним методом спряжених градієнтів. Другий результат означає, що для задачі мінімізації опуклої двічі неперервно-диференційовної функції $f(x)$ граничний варіант r-алгоритмів з відновленням володіє квадратичною швидкістю збіжності за деяких умов гладкості та регулярності $f(x)$. Третій результат полягає в тому, що за певних умов $r_\mu(\alpha)$-алгоритм збігається до точки мінімуму для кусково-гладких опуклих функцій. Що це за результати та якими умовами вони визначаються, буде показано далі.

У граничному варіанті r-алгоритмів припускається, що коефіцієнт розтягу простору нескінченний (при цьому $\beta_k = 0$, $k = 0,1,\ldots$), а крок обирається з умов мінімуму функції $f(x)$ за напрямком антисубградієнта ($h_k = k_k^*, k = 0,1,\ldots$). Якщо $\beta_k = 0$, то оператор $R_{\beta_k}(\eta_k)$ визначається формулою

$$R_0(\eta_k) = I_n - \eta_k \eta_k^T, \quad \text{де} \quad \eta_k = \frac{B_k r_k}{\|B_k r_k\|}, \quad k = 1,2,\ldots,n. \tag{1.10}$$

Формула (1.10) означає, що оператор $R_0(\eta_i)$ є оператором проектування на підпростір, що є ортогональним вектору η_i. Добуток операторів $\prod_{i=1}^{k} R_0(\eta_i)$ не залежить від порядку співмножників, є самоспряженим оператором і виконує проекцію на підпростір, що є ортогональним доповненням до лінійної оболонки взаємно ортогональних векторів η_i, $i = 1,2,\ldots,k$. Звідси випливає така теорема.

Теорема 1.1. Для граничного варіанта r-алгоритмів на деякій ітерації $k^* \le n$ обов'язково виконується умова $B_{k^*}^T g_f(x_{k^*}) = 0$.

Граничний варіант r-алгоритмів є проективним методом спряжених градієнтів. Для невід'ємно визначеної квадратичної функції він знаходить точку мінімуму x^* за кількістю ітерацій, що не перевищує n-розмірності вектора змінних. Із теореми 1.1 випливає, що після k^* ітерацій для граничного варіанта r-алгоритмів продовжувати обчислення неможливо, оскільки напрямок руху в перетвореному просторі перетворюється на нульовий.

Отже, для мінімізації гладких функцій $f(x)$ у граничному варіанті r-алгоритму після виконання умови $B_{k^*}^T g_f(x_{k^*}) = 0$ необхідно застосовувати «відновлення», тобто після деякої кількості ітерацій, що не перевищує n, потрібно «відновлювати» матрицю B_k, замінюючи її одиничною матрицею I_n. Побудований таким чином алгоритм називають граничним варіантом r-алгоритмів з відновленням. Справедлива наступна теорема.

Теорема 1.2. Нехай функція $f(x)$, визначена в E^n, двічі неперервно-диференційовна в деякому околі S точки мінімуму x^*, причому в цьому околі матриця інших похідних $H(x)$ задовольняє умову

$$\|H(x)-H(x')\| \le L\|x-x'\|, \quad x,x' \in S. \tag{1.11}$$

Крім того, $H(x^*)$ — додатно визначена матриця. Тоді для точки x^* знайдеться такий окіл $S' \subseteq S$, що при $x_0 \in S'$ знайдеться таке число $c>0$, що

$$\|x_n - x^*\| \le c\|x_n - x^*\|^2,$$

де x_n — точка, отримана після n кроків роботи вищенаведеного алгоритму (якщо для деякого $k^* < n$ отримали $B_{k^*}^T g_f(x_{k^*}) = 0$, то припускаємо $x_n = x_{k^*}$).

Отже, для задачі мінімізації опуклої двічі неперервно-диференційовної функції $f(x)$ граничний варіант r-алгоритмів з відновленням матриці B_k після кожних n ітерацій має квадратичну швидкість збіжності за умов гладкості та регулярності $f(x)$, наведених у теоремі 1.2.

Для мінімізації гладких опуклих функцій r-алгоритми займають проміжне місце між методом найшвидшого спуску та алгоритмами квазіньютонівського типу зі змінною метрикою. Якщо $\beta_k = 1$ і $h_k = k_k^*$, то розтяг простору не потрібний, і r-алгоритм переходить у метод найшвидшого спуску. Якщо $\beta_k = 0$ і $h_k = k_k^*$, то отримуємо граничний варіант r-алгоритмів, що є проєктивним методом спряжених градієнтів і для квадратичної опуклої функції збігається не більше, ніж за n ітерацій. Якщо $\beta_k = \beta < 1$ і $h_k = k_k^*$, то отримаємо варіант r-алгоритмів, де відновлення матриці B_k не потрібне, а алгоритм буде збігатися швидше, ніж метод найшвидшого спуску. Цим у значній мірі пояснюється чудова властивість r-алгоритмів, яка полягає в тому, що їх конкретні реалізації показують дуже хороші результати при мінімізації яружних опуклих функцій.

У роботах Н. З. Шора $r_\mu(\alpha)$-алгоритм створювався для мінімізації майже диференційовних функцій [9], клас яких є ширшим, ніж клас опуклих функцій. З огляду на це Б. Мордухович, М. Солодов і М. Тодд у передмові до спецвипуску журналу «Optimization Methods and Software» [10], присвяченому Н. З. Шору, зауважили: *«У 1972 році Н.З. Шор ввів фундаментальне поняття узагальненого диференціала для локально ліпшицевих функцій, який він назвав «множина майже-градієнтів». Цей диференціал визначено як сукупність граничних точок звичайних градієнтів ліпшицевої неперервної функції, яка є майже всюди диференційовною за класичною теоремою Радемахера. Згодом ця гранична множина під назвою B-градієнта та B-якобіана вектор-функцій при розробці негладких версій ньютонівського методу. Варто зазначити, що в тій самій статті 1972 року Шор також ввів та використав поняття опуклої оболонки множини майже-градієнтів, яку він назвав «множиною узагальнених майже-градієнтів». Ця множина була перевідкрита Ф. Кларком і стала широко відомою в недиференційовній оптимізації як узагальнений градієнт Кларка для ліпшицевих функцій».*

Результати про збіжність $r_\mu(\alpha)$-алгоритма для мінімізації майже-диференційовних функцій детально викладені в роботах [8] та [4, с. 102–113]. Тому далі опишемо тільки те, до чого приводять ці результати для задачі мінімізації кусково-гладких функцій вигляду

$$f(x)=\max_{1\le i\le m} f_i(x), \quad \text{де} \quad f_i(x)- \text{ гладкі опуклі функції}, \quad i=1,2,\ldots,m. \qquad (1.12)$$

Функція (1.12) опукла. Скористаємося для її мінімізації $r_\mu(\alpha)$-алгоритмом. Позначимо $G_f(x)=\left\{\bigcup_{i\in I(x)} g_{f_i}(x)\right\}$, де $I(x)=\left\{i\middle| f_i(x)=f(x)\right\}$, $g_{f_i}(x)$ — градієнт функції $f_i(x)$ у точці x.

Нехай $\alpha>1$ — коефіцієнт розтягу простору; μ — константа, така, що $0\le\mu<1$; x_0 — початкова точка; $g_f(x_0)$ — субградієнт функції f у точці x_0 (не будь-який субградієнт, а обраний із множини $G_f(x_0)$, тобто

$g_f(x_0) \in G_f(x_0))$; B_0 — невироджена $n \times n$-матриця. $r_\mu(\alpha)$-Алгоритмом є ітераційна процедура побудови послідовностей векторів $\{x_k\}_{k=0}^{\infty}$ та матриць $\{B_k\}_{k=0}^{\infty}$, де перехід від k-ї ітерації до $(k+1)$-ї ітерації відбувається за таким правилом:

1. Обчислимо наступну точку виду

$$x_{k+1} = x_k - \rho_k B_k g_{\varphi_k}(y_k), \tag{1.13}$$

де $g_{\varphi_k}(y_k) = B_k^T g_f(x_k)$, а величина кроку ρ_k обирається з умов:

а) на відрізку $[0, \rho_k]$ функція $\varphi_k(\rho) = f(x_{k+1}(\rho))$ не зростає, (1.13.a)

b) існує $g \in G_f(x_{k+1})$ такий, що $\dfrac{\left(B_k^T g\right)^T g_{\varphi_k}(y_k)}{\left\|B_k^T g\right\| \left\|g_{\varphi_k}(y_k)\right\|} \le \mu$. (1.13.b)

2. Перерахуємо матрицю перетворення простору

$$B_{k+1} = B_k R_\beta(\eta_k) = B_k + (\beta - 1)\left(B_k \eta_k\right)\eta_k^T, \tag{1.14}$$

де

$$\eta_k = \frac{B_k^T r_k}{\| B_k^T r_k \|}, \quad r_k = g_f(x_{k+1}) - g_f(x_k), \quad \beta = \frac{1}{\alpha} < 1. \tag{1.15}$$

3. Здійснимо перехід до наступної ітерації з x_{k+1}, B_{k+1} та $g_f(x_{k+1}) = g$.

Зауваження. Якщо обидві частини формули (1.13) помножити зліва на матрицю $A_k = B_k^{-1}$, то отримаємо

$$y_{k+1} = A_k x_{k+1} = A_k x_k - \rho_k g_{\varphi_k}(y_k) = y_k - \rho_k g_{\varphi_k}(y_k).$$

Тож формула (1.13) фактично реалізує крок субградієнтного спуску для функції $\varphi_k(y) = f(B_k y)$, де ρ_k — невід'ємний крок на k-й ітерації (може дорівнювати нулю). При $\mu = 0$ величина ρ_k^* — це крок найшвидшого спуску в напрямку антисубградієнта. Він пов'язаний з h_k^* (кроком найшвидшого спуску в напрямку нормованого антисубградієнта) за формулою $h_k^* = \rho_k^* \left\|B_k^T g_f(x_k)\right\|$.

Введення константи $\mu \geq 0$ дає можливість розглянути алгоритми, в яких пошук мінімуму за напрямком відбувається приблизно і так, щоб наступна точка знаходилася ближче, ніж точка мінімуму функції за напрямком. Зауважимо, що в r-алгоритмах із розділу 1.2 це було неможливим.

Справді, якщо $\mu \geq 0$, то умова (1.13.b) означає обмеження на косинус гострого кута між двома послідовними субградієнтами, за різницею яких реалізується розтяг простору.

Справедлива така теорема.

Теорема 1.3. Нехай $f(x)$ — функція виду (1.12), така, що $\lim\limits_{\|x\|\to\infty} f(x) = +\infty$, і послідовність $\{x_k\}_{k=0}^{\infty}$, що генерується $r_\mu(\alpha)$-алгоритмом, задовольняє умову

$$\lim_{k\to\infty} \|x_{k+1} - x_k\| = 0. \tag{1.16}$$

Якщо x^* — ізольована точка мінімуму, а точка x_0 така, що опукла множина $\{x : f(x^*) \leq f(x) \leq f(x^0)\}$, яка містить x_0 і x^*, не містить, крім x^*, інших точок z, в яких сімейство $G_f(z)$ лінійно залежне, то послідовність $\{x_k\}_{k=0}^{\infty}$ збігається до точки x^*.

Проблема обґрунтування збіжності $r_\mu(\alpha)$-алгоритмів для всього класу опуклих функцій наразі є відкритою. Одна з причин, через яку для негладких функцій важко довести $r_\mu(\alpha)$-алгоритми, пов'язана з неоднозначним вибором антисубградієнта для наступного напрямку руху з точок негладкості, де субградієнти (з множини G_f) лінійно залежні та жоден із антисубградієнтів не є напрямком спадання функції. Для опуклої кусково-лінійної функції такі точки негладкості можуть бути пастками для мінімізуючої послідовності розглянутого варіанта $r_\mu(\alpha)$-алгоритму [11].

У роботі [12] відзначені шляхи для обґрунтування $r_\mu(\alpha)$-алгоритмів. Тут розглядаються монотонні модифікації r-алгоритмів, для яких за критерій зупинки обрана необхідна та достатня умова оптимальності для опуклих функцій — $0 \in \partial f(x)$, де $\partial f(x)$ – субдиференціал. $r_\mu(\alpha)$-Алгоритм доповнюєть-

ся таким правилом. Якщо в певній точці мінімум за напрямком реалізується при нульовому кроці, то для наступної ітерації напрямок руху в перетвореному просторі обирається аналогічно найшвидшому спуску для опуклих функцій.

Модифікований вищенаведеним способом $r_\mu(\alpha)$-алгоритм для майже диференційовних кусково-гладких функцій завжди гарантує вихід з будь-якої точки, в якій існує напрямок спадання функції, і, крім того, дає можливість стверджувати, що отримано оптимум, якщо найкоротшим до опуклої оболонки майже-градієнтів виявиться нульовий вектор.

Така схема приваблива в теоретичному плані, оскільки питання збіжності при цьому фактично зводяться до визначення існування в точці напрямку спадання функції. Але їх так само, як і $r_\mu(\alpha)$-алгоритм, можна вважати «ідеалізованими», оскільки для реалізації найшвидшого спуску з високою точністю необхідно виконати велику кількість обчислень значень функції та її субградієнта.

У практичних варіантах r-алгоритмів крок h_k обирається так, щоб виконувалася нерівність $h_k > h_k^*$, де h_k^* відповідає мінімуму функції. Для них «наближений» пошук мінімуму функції спрямований на зменшення загальної кількості обчислень значень функції та її субградієнта і реалізується таким чином, щоб на одну ітерацію алгоритму припадало в середньому два-три таких обчислення. Однією з ефективних реалізацій такої стратегії є $r(\alpha)$-алгоритм з адаптивним регулюванням кроку.

1.4. $r(\alpha)$-Алгоритм з адаптивним кроком та його програмні реалізації

Одним з ефективних є $r(\alpha)$-алгоритм з адаптивним регулюванням кроку, де α — постійний коефіцієнт розтягу простору, а величина кроку h_k налаштовується під час виконання одночасного спуску в напрямку нормованого антисубградієнта в перетвореному просторі змінних. Налаштування кроку

здійснюється за допомогою чотирьох параметрів: $h_0 > 0$ — величина початкового кроку (використовується на першій ітерації, на кожній наступній уточнюється); q_1 — коефіцієнт зменшення кроку ($q_1 \le 1$), якщо умова завершення спуску за напрямком виконується за один крок; q_2 — коефіцієнт збільшення кроку ($q_2 \ge 1$); через кожні n_h кроків одномірного спуску ($n_h > 1$) величина кроку збільшуватиметься у q_2 разів. Умова завершення спуску за напрямком виконується, як тільки буде знайдена точка x_{k+1}, для якої

$$\left(x_{k+1} - x_k\right)^T g_{k+1}(x_{k+1}) \ge 0. \tag{1.17}$$

Умова (1.17) легко перевіряється, оскільки в силу додатності кроку вона рівносильна виконанню нерівності

$$\left(B_k B_k^T g_f(x_k)\right)^T g_f(x_{k+1}) \le 0. \tag{1.18}$$

Це означає, що кут між двома послідовними субградієнтами в перетвореному просторі змінних буде негострим. Нерівність (1.18) можна записати як умову

$$\left(B_k B_k^T g_f(x_k)\right)^T g_f(x_{k+1}) \le 0, \tag{1.19}$$

звідки

$$\left(g_\varphi(y_k)\right)^T g_\varphi(y_{k+1}) \le 0, \tag{1.20}$$

де $g_\varphi(y_k) = B_k^T g_f(x_k)$ і $g_\varphi(y_{k+1}) = B_k^T g_f(x_{k+1})$ є субградієнтами функції $\varphi(y) = f(B_k y)$ в точках $y_k = A_k x_k$ и $y_{k+1} = A_k x_{k+1}$ перетвореного простору змінних $y = A_k x$, де $A_k = B_k^{-1}$. Оскільки передбачається, що $\lim\limits_{\|x\| \to \infty} f(x) = +\infty$, то після скінченої кількості кроків адаптивного спуску в напрямку нормованого антисубградієнта обов'язково виконуватиметься умова завершення (1.19)–(1.20).

Ітеративний процес в $r(\alpha)$-алгоритмі з адаптивним регулюванням кроку продовжується до виконання деякого критерію зупинки, де визначальну роль відіграють параметри ε_x та ε_g. Алгоритм зупиняється в точці x_{k+1}, якщо виконано умову $\left\|x_{k+1} - x_k\right\| \le \varepsilon_x$ (зупинка за аргументом) або умову

$\left\| g_f(x_{k+1}) \right\| \le \varepsilon_g$ (зупинка за нормою субградієнта, використовується для гладких функцій). Крім того, використовуються ще дві умови зупинки: стандартна зупинка, якщо перевищено максимальну кількість ітерацій, та аварійна зупинка, яка сигналізує про те, що або функція $f(x)$ необмежена знизу, або початковий крок h_0 занадто малий і його необхідно збільшити. І хоча $r(\alpha)$-алгоритм з адаптивним регулюванням кроку не гарантує монотонного спадання функції, проте, як показали експерименти, зростання функції відбувається достатньо рідко.

Адаптивний спосіб регулювання кроку дозволяє збільшувати точність пошуку мінімуму функції в процесі розрахунку, щоб кількість кроків за напрямком не перевищувала в середньому двох-трьох на одну ітерацію [13].

Якщо $r(\alpha)$-алгоритм з адаптивним регулюванням кроку застосувати для мінімізації негладких функцій, то рекомендується такий вибір параметрів: $\alpha = 2 \div 4$, $h_0 = 1.0$, $q_1 = 1.0$, $q_2 = 1.1 \div 1.2$, $n_h = 2 \div 3$. Якщо відома оцінка відстані від початкової точки x_0 до точки мінімуму x^*, то початковий крок h_0 доцільно обирати близьким до $\| x_0 - x^* \|$. При мінімізації гладких функцій рекомендовані такі самі параметри, за винятком q_1 ($q_1 = 0.8 \div 0.95$).

Це зумовлене тим, що додаткове зменшення кроку сприяє збільшенню точності пошуку мінімуму функції за напрямком, що при мінімізації гладких функцій дає вищу швидкість збіжності. При такому виборі параметрів, як правило, число спусків за напрямком рідко буде більшим двох, а за n кроків точність по функції поліпшується у три-п'ять разів.

Параметри зупинки $\varepsilon_x, \varepsilon_g \sim 10^{-6} \div 10^{-5}$ при мінімізації опуклої функції навіть суттєво яружної структури забезпечують знаходження точки x_k^* – наближення до точки x^*, для якої значення функції достатньо близьке до оптимального ($\frac{f(x_k^*) - f(x^*)}{|f(x^*)|+1} \sim 10^{-6} \div 10^{-5}$ для негладких і $\frac{f(x_k^*) - f(x^*)}{|f(x^*)|+1} \sim 10^{-12} \div 10^{-10}$ для

гладких функцій). Це підтверджується результатами чисельних тестових і реальних розрахунків.

У наш час $r(\alpha)$-алгоритм з адаптивним регулюванням кроку і його модифікації реалізовані багатьма комп'ютерними програмами на мовах програмування Фортран, C, C++, C# і Octave. В їх основу покладені B-форми r-алгоритму. За допомогою економної B-форми (1.5)–(1.7) і її модифікацій роз-роблені комп'ютерні програми **ralg** (Фортран, C, C++), **ralgb4** (Фортран, Octave), **SolveOpt** (Фортран, C). На основі B-форми методу (1.1)–(1.3) розроблена комп'ютерна програма **ralgb5** (Фортран, Octave, C++ і C#). Для r-алгоритмів у H-формі комп'ютерні програми не отримали широкої практики в силу того, що виявилися неефективними для достатньо точного вирішення задач оптимізації.

Історично однією з перших була фортранівська програма **ralg** (автор – М. Г. Журбенко). У 70–80 роках минулого століття програма **ralg** активно використовувалася для оптимізації негладких функцій в Інституті кібернетики та інших організаціях. Наприклад, активними користувачами програми були Д. І. Соломон (м. Кишинів, Інститут математики) і О. М. Кісєльова (Дніпропетровський університет), які захистили докторські дисертації під керівництвом Н. З. Шора. У 1990-ті роки програма **ralg** стала основою для фортранівської програми **ralgb4** (автор — П. І. Стецюк), в якій використовується модифікація r-алгоритму [14]. За допомогою програми **ralg** О. В. Кунцевич розробив комплекс програм **SolveOpt** (мови — Фортран та C), де використане ускладнене адаптивне регулювання кроку та критерії зупинки $|x_{k+1}^i - x_k^i| \le \delta_x |x_{k+1}^i|$ і $|f(x_{k+1}) - f(x_k)| \le \delta_f |f(x_{k+1})|$ із заданими достатньо малими δ_x і δ_f [15].

В кінці минулого – на початку цього століття програма **ralg** стала основою для розробки комп'ютерних програм на мовах програмування C, C++ (автори – М. Г. Журбенко і О.П. Лиховид).

У 2007–2008 роках була розроблена фортранівська програма **ralgb5** (автор — П. І. Стецюк), в якій використовується метод (1.1)–(1.3). Її назва пов'язана з тим, що в основу програми закладена B-форма r-алгоритму, яка потребує $5n^2$ арифметичних операцій множення на кожній ітерації. У 2010 році програма **ralgb5** була переписана мовою Octave (її код наведено в [16, с. 384–385]). При розв'язанні задач для тисячі і більше змінних Octave-програма **ralgb5** виявилася швидшою за однойменну фортранівську програму. Це обумовлено тим, що бібліотека BLAS (Basic Linear Algebra Subprograms) для мови Octave дозволяє швидше виконувати матрично-векторні операції в r-алгоритмах, ніж це дозволяють оптимізуючі опції компілятора Фортрана. На сьогодні програма **ralgb5** реалізована О.П. Лиховидом на мовах C++, C# і використовується в програмних імплементаціях алгоритмів вирішення різних задач нелінійного програмування.

У 2016 році по аналогії з програмою **ralgb5** була розроблена Octave-програма **ralgb4** [17]. Вона використовує метод (1.5)–(1.7), який реалізує економну B-форму r-алгоритмів. Octave-програма **ralgb4** потребує $4n^2$ арифметичних операцій множення на кожній ітерації. Її обчислювальні властивості виявилися близькими до обчислювальних властивостей програми **ralgb5.** Octave-функції **ralgb4** і **ralgb5** можна використовувати як оптимізаційні ядра при реалізації на мові **Octave** алгоритмів вирішення задач нелінійного програмування. На їх основі легко розробляти оптимізаційні ядра мовою **MATLAB** для вирішення обчислювальних задач, які зводяться до проблем мінімізації негладких випуклих функцій або гладких випуклих функцій з яружною структурою поверхонь рівня.

Octave-функції **ralgb4** і **ralgb5** можна легко переписати на мовах **Фортран** і **C**, 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 дозволить значно пришвидшити методи для

розв'язання громіздких задач (тисяча і більше змінних), наприклад, за рахунок використання обчислювальних потужностей графічних процесорів, які в декілька разів перевищують обчислювальні потужності класичних процесорів. Так, значне прискорення можна отримати використовуючи для розрахунків технологію CUDA на графічних прискорювачах. Це підтверджує гібридна реалізація r-алгоритму [18], де скорочення часу, яке досягається при розв'язанні задач з 1000–8000 змінними, варіюється від 14 до 18 разів.

За допомогою програмних реалізацій $r(\alpha)$-алгоритму з адаптивним кроком можна знаходити досить точні наближення до точки мінімуму опуклої функції. Якщо коефіцієнт розтягу простору вибрати таким, щоб він добре узгоджувався з параметрами адаптивного регулювання кроку в напрямку нормованого антисубградієнта в перетвореному просторі змінних, то для виконання одних і тих самих критеріїв зупинки можна значно скоротити кількість ітерацій і кількість обчислень значення функції і субградієнта (градієнта). Це залежить від конкретного виду функції, що мінімізується, ступеня її яружності та масштабу змінних.

1.5. Octave-функція ralgb5a

Про програму ralgb5a [19, 20]. Програма **ralgb5a** є спрощеною (для зручності використання) версією програми **ralgb5** [16, с. 383–386], в якій використо-вується метод (1.1)–(1.3) [3]. Тут абревіатура «**b5»** означає, що в основу програми покладено r-алгоритм у B-формі, де коректується $n \times n$-матриця B, а кожна ітерація методу (1.1)–(1.3) вимагає $5n^2$ арифметичних операцій множення, які визначають обчислювальну трудоємність ітерації (операції додавання не враховуються через їх малий внесок у трудоємність ітерації). З них $3n^2$ операцій множення потрібно для обчислення векторів $B_k\xi_k$, $B_k^T g_f(x_k)$ і $B_k^T r_k$ (множення матриці на вектор), а $2n^2$ операцій множення

вимагає однорангова корекція матриці $B_{k+1} = B_k R_\beta(\eta_k)$. Дійсно, корекція матриці B_{k+1} виконується за формулою

$$B_{k+1} = B_k R_\beta(\eta_k) = B_k(I_n + (\beta - 1)\eta_k \eta_k^T) = B_k + (\beta - 1)(B_k \eta_k)\eta_k^T,$$

звідки легко бачити, що обчислення вектора $\eta = B_k \eta_k$ вимагає n^2 операцій множення, і стільки ж операцій множення вимагає побудова тимчасової однорангової матриці $\eta \eta_k^T$.

У програмі **ralgb5a** зафіксовані два найбільш часто використовувані параметри $q_2 = 1.1$ і $n_h = 3$. При цьому величина початкового кроку для чергової ітерації може максимально збільшуватися в 10^6 разів. У програмі **ralgb5a** використовується параметр **intp (int**erval for **p**rint**)**, який забезпечує друк інформації про хід процесу мінімізації через кожні **intp** ітерацій. Цей параметр дозволяє скоротити протокол роботи програми при мінімізації функції для сотень і тисяч змінних, коли кількість ітерацій оцінюється тисячами і десятками тисяч. Програма використовує параметри ε_x і ε_g для зупинки ітераційного процесу в точці $x_{k^*} \in [x_k, x_{k+1}]$, де $\|x_{k+1} - x_k\| \le \varepsilon_x$ (зупинка за аргументом) або $\|g_f(x_{k^*})\| \le \varepsilon_g$ (зупинка за нормою градієнта, яка використовується для гладких функцій). Використовуються також стандартна зупинка, якщо перевищено задану максимальну кількість ітерацій **maxitn**, та аварійна зупинка, яка сигналізує про те, що або функція $f(x)$ не є обмеженою знизу, або початковий крок h_0 занадто малий, і його треба збільшити.

Якщо ітераційний процес запускається зі стартової точки x_0, то параметри $r(\alpha)$-алгоритму рекомендується вибирати такими: $\alpha \in [2,4]$, $q_1 = 1.0$ (для негладких функцій), $q_1 = 0.8 \div 0.95$ (для гладких функцій), $h_0 \approx \|x_0 - x^*\|$ — оцінка відстані від стартової точки x_0 до точки мінімуму x^*. Як правило, використовуються такі параметри зупинки: $\varepsilon_x \approx 10^{-6}$; $\varepsilon_g \approx 10^{-12}$; **maxitn** $\approx 20n$**.**

Тут параметр ε_g використовується для гладких функцій, а параметр ε_x — для негладких функцій. Якщо програма **ralgb5a** завершує роботу за умовою $\|x_{k+1} - x_k\| \le \varepsilon_x = 10^{-8}$, то цього цілком достатньо, щоб на $14-15$ порядків зменшити різницю між знайденим рекордним значенням квадратичної функції f_r та її мінімальним значенням f^*.

Опис програми ralgb5a. Нижче наведемо короткий опис програми ralgb5a, який містимите код Octave-функції ralgb5a та його застосування до тестового прикладу sabs(100,1.2), який полягає у мінімізації кусково-лінійної функції

$$f(x) = \sum_{i=1}^{100} (1.2)^{i-1} |x_i - 1|, \quad f^* = f(x^*) = 0, \quad x^* = (1,1,\ldots,1)^T, \tag{1.21}$$

де $|a|$ — абсолютна величина числа a. Функція (1.21) є яружною, оскільки коефіцієнти при $|x_i - 1|$, $i = 1,\ldots,100$ утворюють геометричну прогресію з показником $q = 1.2$, де мінімальний коефіцієнт дорівнює $(1.2)^0 = 1$, а максимальний – $(1.2)^{99} \approx 6.9015\text{e+}07$.

Код програми ralgb5a. Octave-функція ralgb5a знаходить x_r^* — наближення до точки мінімуму опуклої функції $f(x)$ від n змінних. Програма використовує Octave-функцію **function [f, g] = calcfg (x)**, яка обчислює значення функції $f = f(x)$ та її субградієнта $g = g_f(x)$ в точці x. Ця програма готується користувачем та може мати довільне ім'я, яке підтримує синтаксис Octave. Код програми, що включає і короткі англомовні коментарі для вхідних та вихідних параметрів, наведено нижче.

```
# Input parameters:
#    calcfg - name of the function calcfg(x) for calculation of f and g
#    x - the starting point, x0(1:n) (it is modified in the program)
#    alpha - the value of coefficient of space dilation
#    h0, q1 - parameters of the adaptive step adjustment
#    epsx, epsg, maxitn - stop parameters
#    intp - print information every intp iteration
# Output parameters:
```

```
#    xr - a minimum point, which was found by the program, xr(1:n)
#    fr - the value of the function f at the point xr
#    itn - the number of iterations used by the program
#    nfg - the number of function calcfg calls
#    istop - exit code (2 = epsg, 3 = epsx, 4 = maxitn, 5 = error)
function [xr,fr,itn,nfg,istop] = ralgb5a(calcfg,x,alpha,h0,q1,  #row001
                                         epsg,epsx,maxitn,intp);
itn = 0; B = eye(length(x)); hs = h0; lsa = 0; lsm = 0;         #row002
xr = x; [fr,g0] = calcfg(xr); nfg = 1;                          #row003
printf("itn %4d  f%15.6e  fr%15.6e  nfg %4d\n",itn,fr,fr,nfg);  #row004
if(norm(g0) < epsg) istop = 2;  return; endif                   #row005
for (itn = 1:maxitn)                                            #row006
   dx = B * (g1 = B' * g0)/norm(g1);                            #row007
   d = 1; ls = 0; ddx = 0;                                      #row008
   while (d > 0)                                                #row009
      x -= hs * dx; ddx += hs * norm(dx);                       #row010
      [f, g1] = calcfg(x); nfg ++;                              #row011
      if (f < fr)  fr = f; xr = x;  endif                       #row012
      if(norm(g1) < epsg) istop = 2;  return; endif             #row013
      ls ++; (mod(ls,3) == 0) && (hs *= 1);                     #row014
      if(ls > 500) istop = 5;  return; endif                    #row015
      d = dx' * g1;                                             #row016
   endwhile                                                     #row017
   (ls == 1) && (hs *= q1);  lsa=lsa+ls; lsm=max(lsm,ls);       #row018
   if(mod(itn,intp)==0)                                         #row019
     printf("itn %4d  f %14.6e  fr %14.6e", itn, f, fr);        #row020
     printf("  nfg %4d  lsa %3d  lsm %3d\n", nfg, lsa, lsm);    #row021
     lsa=0; lsm=0;                                              #row022
   endif                                                        #row023
   if(ddx < epsx) istop = 3;  return; endif                     #row024
   xi = (dg = B' * (g1 - g0) )/norm(dg);                        #row025
   B += (1 / alpha - 1) * B * xi * xi';                         #row026
   g0 = g1;                                                     #row027
endfor                                                          #row028
istop = 4;                                                      #row029
endfunction                                                     #row030
```

При мінімізації негладких функцій рекомендується вибирати: alpha = $2 \div 3$, $h0 = 0$, $q1 = 0$. При мінімізації гладких функцій рекомендується використовувати $q1 = 0.8 \div 0.95$. За правильного підбору цих параметрів можна значно скоротити кількість ітерацій для виконання одних і тих самих критеріїв зупинки. Це залежить від конкретного виду функції, що мінімізується, ступеня її яружності та масштабу змінних.

Тестовий приклад. Для тестового прикладу використовується кусково-лінійна функція (1.21). Обчислення значення функції (1.21) та її субградієнта реалізовано Octave-функцією

```
function [f,g] = sabs(x)
global w
temp=x-ones(length(x),1); f=sum(abs(w.*temp)); g=w.*sign(temp);
endfunction,
```

для якої значення коефіцієнтів w встановлюються за допомогою операторів

```
global w
n=100; temp=[0:(n-1)]'; w=1.2**temp.
```

Ітераційний процес запускається зі стартової точки $x_0 = (0,0,\ldots,0)^T$, для якої значення функції $f(x_0) = 4.140899\text{e+}08$. Параметри $r(\alpha)$-алгоритма вибираються такими: $\alpha = 4$ (рекомендується $\alpha \in [2,4]$), $q_1 = 1.0$ (рекомендується для негладких функцій), $h_0 = 10$ (дорівнює $\|x_0 - x^*\|$ — відстані від стартової точки x_0 до точки мінімуму x^*). Використовуються параметри зупинки: $\varepsilon_x = 10^{-8}$, $\varepsilon_g = 10^{-12}$, **maxitn = 5000**. Тут параметр ε_g ролі не відіграє (він використовується для гладких функцій), а параметр ε_x вибраний таким, щоб программа **ralgb5a** закінчувала роботу за критерієм $\|x_{k+1} - x_k\| \le 10^{-8}$, чого цілком достатньо, щоб на 14–15 порядків зменшити різницю між знайденим рекордним значенням функції f_r та її мінімальним значенням $f^* = 0$.

Для вказаних параметрів головна Octave-програма має вигляд:

```
global w
n=100; temp=[0:(n-1)]'; w=1.2**temp; # w(1,1), w(100,1),
x = zeros(n,1); alpha = 4.0, h0 = 10.0, q1 = 0,
epsx = e-8, epsg = e-12, maxitn = 5000, intp=500;
[xr,fr,itn,nfg,istop]=ralgb5a(@sabs,x,alpha,h0,q1,epsg,epsx,maxitn,intp);
printf("itn %4d  fr %23.15e  istop %d  nfg %4d\n", itn, fr, istop,nfg);
dx = norm(xr-ones(n,1)),
```

Протокол роботи програми. Обчислення проводилися на комп'ютері Pentium 3GHz у системі Windows7/32 за допомогою GNU Octave версії 3.6.4. Протокол роботи програми **ralgb5a** при **intp = 500** має такий вигляд**:**

```
alpha =  4  h0 =  10  q1 =  1
epsx = 0000e-008  epsg = 0000e-012  maxitn =  5000
itn    0  f  4.140899e+008  fr  4.140899e+008  nfg    1
itn  500  f  718525e+003  fr  273433e+003  nfg  532  lsa 531  lsm   4
itn 1000  f  409472e+000  fr  192802e+000  nfg 1032  lsa 500  lsm   1
itn 1500  f  258921e-003  fr  258921e-003  nfg 1532  lsa 500  lsm   1
itn 2000  f  422859e-006  fr  224438e-006  nfg 2032  lsa 500  lsm   1
itn 2046  fr  6.340398755873688e-007  istop 3  nfg 2078
dx = 9497e-008
```

Тут вхідні параметри $r(\alpha)$-алгоритма зібрано у двох перших рядках. Із протоколу видно, що кількість ітерацій відповідає емпіричній оцінці, тобто на одну ітерацію в середньому затрачено $2078/2046<3$ обчислень значення функції (1.21) та її субградієнта. Якщо вибрати параметр $q_1=0.95$, то кількість ітерацій зменшується до **920** при **1539** викликах функції **sabs**.

Програма **ralgb5a** працює під управлінням тих операційних систем, які допускають установку Open Source-пакета для математичних обчислень GNU Octave [21]. Програма використовує версії Octave 3.0.0 і вище. Для неї не потрібно ніяких спеціальних конфігурацій комп'ютера.

1.6. Висновки

У рамках сімейства субградієнтних методів з розтягом простору в напрямку різниці двох послідовних субградієнтів отримані достатньо ефективні реалізації r-алгоритмів. Розроблені модифікації r-алгоритмів можна використовувати при мінімізації опуклих негладких функцій з різних сфер застосувань. Оскільки гладка функція з градієнтом, що дуже швидко змінюється, близька за своїми властивостями до негладкої функції, то r-алгоритми мають прискорену збіжність при оптимізації яружних гладких функцій. При мінімізації гладких

функцій вони виявляються конкурентоспроможними з найбільш вдалими реалізаціями методів спряжених напрямків і методів квазіньютонівського типу.

За допомогою програмних реалізацій $r(\alpha)$-алгоритму з адаптивним кроком можна знаходити достатньо точні наближення до точки мінімуму опуклої функції. Якщо коефіцієнт розтягу простору обрати таким, щоб він добре узгоджувався з параметрами адаптивного регулювання кроку в напрямку нормованого антисубградієнта в перетвореному просторі змінних, то для виконання одних і тих самих критеріїв зупинки можна значно скоротити кількість ітерацій і кількість обчислень функції та суградієнта (градієнта). Це залежить від конкретного виду мінімізуючої функції, ступеня її яружності та масштабу змінних [22, 23].

Однак теоретичне обґрунтування r-алгоритмів проведене недостатньо повно. І хоча кількість ітерацій для знаходження мінімального значення f^* з точністю ε для опуклих функцій від n змінних емпірично оцінюється як $N = O\left(n \log \frac{1}{\varepsilon}\right)$, наразі актуальною виявилася цитата Н. З. Шора 1982 року: «Теория всего класса алгоритмов с растяжением пространства далека от совершенства. Нам кажется достаточно реалистичной целью построение такого алгоритма, который по своей практической эффективности не уступал бы *r*-алгоритму и был столь же хорошо обоснован, как метод эллипсоидов».

Дослідження в цьому напрямку активно тривають.

Список літератури

1. Шор Н. З. Методы минимизации недифференцируемых функций и их приложения: Автореф. дис. докт. физ-мат. наук. Киев, 1970. 44 с.
2. Шор Н. З., Журбенко Н. Г. Метод минимизации, использующий операцию растяжения пространства в направлении разности двух последовательных градиентов. Кибернетика. 1971. № 3. С. 51–59.

3. Стецюк П. И. Теория и программные реализации r-алгоритмов Шора. Кибернетика и системный анализ. 2017. № 5. С. 43–57.
4. Шор Н. З. Методы минимизации недифференцируемых функций и их приложения. Київ: Наукова думка, 1979. 200 с. (English transl.: Shor N. Z. Minimization Methods for Non-Differetiable Functions. Berlin: Springer-Verlag, 1985. 178 p.)
5. Shor N. Z. Nondifferentiable optimization and polynomial problems. Boston; Dordrecht; London: Kluwer Academic Publishers. 1998. 412 p.
6. Shor N. Z., Stetsyuk P. I. Lagrangian bounds in multiextremal polynomial and discrete optimization problems. Journal of Global Optimization. 2002. Vol. 23, P. 1–41.
7. Шор Н. З., Журбенко Н. Г., Лиховид А. П., Стецюк П. И. Развитие алгоритмов недифференцируемой оптимизации и их приложения. Кибернетика и системный анализ. 2003. № 4. С. 80–94.
8. Шор Н. З. Исследование сходимости метода градиентного типа с растяжением пространства в направлении разности двух последовательных субградиентов. Кибернетика. 1975. № 4. С. 48–53.
9. Шор Н. З. О классе почти-дифференцируемых функций и одном методе минимизации функций этого класса. Кибернетика. 1972. № 4. С. 65–70.
10. Special issue «Nonsmooth optimization and related topics», dedicated to the memory of professor Naum Shor, Optimization Methods and Software, 23:1, 3–4 (2008) DOI: 10.1080/10556780701652368.
11. Стецюк П. И. К вопросу сходимости r-алгоритмов. Кибернетика и системный анализ. 1995. № 6. С. 173–177.
12. Шор Н. З. Монотонные модификации r-алгоритмов и их приложения. Кибернетика и системный анализ. 2002. № 6. С. 74–96.
13. Шор Н. З., Стеценко С. И. Квадратичные экстремальные задачи и недифференцируемая оптимизация. Киев: Наук. думка, 1989. 208 с.

14. 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.
15. Kappel F., Kuntsevich A. V. An implementation of Shor's r-algorithm. Computational Optimization and Applications. 2000. Vol. 15, № 2. P. 193–205.
16. Стецюк П. И. Методы эллипсоидов и r-алгоритмы. Кишинэу: Эврика, 2014. 488 с.
17. Стецюк П. И. Субградиентные методы ralgb5 и ralgb4 для минимизации овражных выпуклых функций. Вычислительные технологии. 2017. № 2. С. 127–149.
18. Стецюк П. І., Хіміч О. М., Сидорук В. О. Реалізація r-алгоритму на графічних процесорах. Компьютерная математика. Киев: Ин-т кибернетики имени В. М. Глушкова НАН Украины, 2016. № 2. С. 100–109.
19. Стецюк П. И., Фишер А. r-Алгоритмы Шора и octave-функция ralgb5a. Тези Міжнародної наукової конференції «Сучасна інформатика: проблеми, досягнення та перспективи розвитку», присвяченої 60-річчю заснування Інституту кібернетики імені В. М. Глушкова НАН України (м. Київ, 13–15 грудня 2017 р.). Київ: Ін-т кібернетики імені В. М. Глушкова НАН України, 2017. С. 143–146.
20. 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 освіти і науки. Державний департамент інтелектуальної власності. Дата реєстрації 29.01.2019.
21. Octave. URL: http://www.octave.org.
22. 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 S. Butenko, P. M. Pardalos, V. Shylo. Springer. 2017. P. 495–520.
23. Лиховид А. П., Ивличев А. В. Результаты тестирования двух реализаций r-алгоритма. Теорія оптимальних рішень. Київ: Ін-т кібернетики імені В. М. Глушкова НАН України, 2018. С. 42–48.

2. СУБГРАДІЄНТНІ МЕТОДИ З КРОКОМ ПОЛЯКА ТА ПРОГРАМА AMSG2P

П. І. Стецюк, В. О. Стовба

Анотація. Розглядаються три субградієнтні методи (методи **A** та **B**, метод **amsg2p**) для знаходження точки мінімуму опуклої функції з відомим оптимальним значенням функції та їхні програмні реалізації мовою Octave: метод **A** — субградієнтний метод, що використовує крок Поляка в початковому просторі змінних (програма **PolyakA**); метод **B** — субградієнтний метод, що використовує крок Поляка в перетвореному просторі змінних (програма **PolyakB**); **amsg2p** — субградієнтний алгоритм з кроком Поляка та перетворенням простору за допомогою двох послідовних субградієнтів та агрегатного вектора (програма **amsg2p**).

Annotation. Three subgradient methods (methods **A** and **B**, method **amsg2p**) to find the minimum point of a convex function with known optimal values of a function and their program implementations in Octave are considered: method A is a subgradient method using Polyak's step in the original space of variables (program **PolyakA**); method **B** is a subgradient method using Polyak's step in the transformed space of variables (program **PolyakB**); **amsg2p** is a subgradient algorithm with Polyak's step and transformation of space using two successive subgradients and the aggregate vector (program **amsg2p**).

2.1. Вступ

Розглядаються три субградієнтні методи (методи **A** та **B** [1, 2], метод **amsg2p**) для знаходження точки мінімуму опуклої функції з відомим оптимальним значенням функції.

Метод **A** — це субградієнтний метод, що використовує крок Поляка в початковому просторі змінних. Метод **B** — це субградієнтний метод у перетвореному просторі змінних, що використовує крок Поляка в перетвореному просторі. Для обох методів розглядаються доведення збіжності до точки мінімуму із заданою точністю, що визначається оптимальним значенням функції. Наведені приклади гладких та негладких яружних опуклих

функцій, для яких метод **A** збігається повільно. Показано, що правильно обравши матрицю перетворення, метод **B** можна суттєво прискорити в порівнянні з методом **A** для яружних опуклих функцій.

Крок Поляка використовується також у методі **amsg2p**, де перетворення простору направлене на зменшення яружності функції в черговому просторі змінних. Перетворення реалізується за допомогою однорангового еліпсоїдального оператора на тих ітераціях методу, коли тупим є хоча б один із кутів — або кут між двома послідовними субградієнтами, або кут між останнім субградієнтом та агрегатним вектором, який є опуклою комбінацією обчислених на попередніх ітераціях субградієнтів.

2.2. Позначення та визначення

Нехай $f(x)$ — опукла функція, $x \in R^n$. Позначимо її мінімальне значення як $f^* = f(x^*)$ та припустимо, що точка мінімуму x^* — єдина. Субградієнт $g_f(x)$ задовольняє таку умову:

$$(x - x^*, g_f(x)) \geq f(x) - f^*, \ \forall x \in R^n. \tag{2.1}$$

Тут (x, y) — скалярний добуток векторів $x \in R^n$ та $y \in R^n$.

Якщо $f(x)$ є неперервно-диференційовною в точці $\bar{x}$, то субградієнт $g_f(\bar{x})$ однозначно визначається і збігається з $\nabla f(\bar{x})$ — градієнтом функції $f(x)$ у точці $\bar{x}$. У точках, де функція $f(x)$ негладка, субградієнт $g_f(\bar{x})$ визначається неоднозначно.

Нерівність (2.1) випливає з визначення субградієнта $g_f(x)$ для опуклої функції $f(x)$. Справді, субградієнт $g_f(x)$ у точці x задовольняє нерівність

$$f(y) - f(x) \geq (g_f(x), y - x), \ \forall y \in R^n, \tag{2.2}$$

яка виконується також для x^* — точки мінімуму. Для точки x^* нерівність (2.2) перетворюється на нерівність

$$f(x^*) - f(x) \geq (g_f(x), x^* - x),$$

яку, зважаючи на те, що $f(x^*) = f^*$, можна записати як нерівність (2.1).

Якщо f^* відоме, то для знаходження наближення до точки x^* можна використати субградієнтний метод Поляка [3]:

$$x_{k+1} = x_k - h_k \frac{g_f(x_k)}{\|g_f(x_k)\|}, \; h_k = \frac{f(x_k) - f^*}{\|g_f(x_k)\|}, \; k = 0,1,2... \qquad (2.3)$$

Крок h_k називається кроком Поляка (або кроком Агмона – Моцкіна – Шонберга). Вперше крок Поляка використовувався для мінімізації кусково-лінійних опуклих функцій. Так, у 1954 році Агмон [4], Моцкін і Шонберг [5] використали цей крок у релаксаційному методі для знаходження хоча б одного розв'язку сумісної системи лінійних нерівностей. У 1965 р. І. І. Єрьомін [6] узагальнив цей релаксаційний метод для системи опуклих функцій.

Геометричний зміст методу (2.3) є таким. Функція $f(x)$ апроксимується лінійною функцією $\widetilde{f}(x) = f(x_k) + (g_f(x_k), x - x_k)$, і крок вибирається таким чином, щоб апроксимуюча функція стала рівною f^* (тобто $\widetilde{f}(x_{k+1}) = f^*$). Для опуклої функції $f(x)$ крок h_k визначає таку величину максимального зміщення з точки x_k у напрямку нормалізованого антисубградієнта, для якого умова (2.1) гарантує, що кут між антисубградієнтом у точці x_k та напрямком від точки x_{k+1} до точки мінімуму x^* не буде тупим.

Це означає такий факт. Субградієнт у точці x_k визначає гіперплощину, яка локалізує точку x^* у півпросторі в напрямку антисубградієнта. Якщо її перемістити на величину максимального зміщення у напрямку нормованого анти-субградієнта, то нова гіперплощина локалізуватиме x^* у такому півпросторі щодо точки x_{k+1}.

Далі розглянемо субградієнтний метод Поляка (метод **A**, підрозділ 2.3) та його прискорені варіанти (метод **B**, підрозділ 2.4, метод **amsg2p**, підрозділ 2.5)

для знаходження наближення до точки мінімуму яружних опуклих функцій. Як критерій зупинки використаємо умову $f(x_k) - f^* \le \varepsilon$; для довільного малого $\varepsilon > 0$ це дозволяє нам знайти таку точку $x_\varepsilon^* = x_k$, що $f(x_\varepsilon^*) \le f^* + \varepsilon$.

Розглядатимемо методи **A** та **B** для більш загального випадку опуклої функції $f(x)$, коли її субградієнт $g_f(x)$ задовольняє таку умову:

$$(x - x^*, g_f(x)) \ge m(f(x) - f^*), \forall x \in R^n, \tag{2.4}$$

де параметр $m \ge 1$. Параметр m вводиться для того, щоб урахувати спеціальні класи опуклих функцій; наприклад, для кусково-лінійних негладких функцій $m = 1$, для квадратичних гладких $m = 2$, для функцій виду $f(x) = \sum_{i=1}^{k} \left| \sum_{j=1}^{n} a_{ij} x_j - b_i \right|^p$, де $p > 1$, $m = p$.

Параметр m можна активно використовувати для диференційованих однорідних з показником σ опуклих функцій. Для них виконується рівність $\sigma\left(f(x) - f^*\right) = \left(x - x^*, g_f(x)\right)$, отже параметр m можна вибрати $m = \sigma > 1$.

2.3. Субградієнтний метод Поляка

Опис методу А. Якщо опукла функція $f(x)$ задовольняє умову (2.4) та f^* відоме, тоді для знаходження точки $x_\varepsilon^* \in R^n$ такої, що $f(x_\varepsilon^*) \le f^* + \varepsilon$, можна використати такий ітеративний метод.

Ініціалізація. Нехай f^* та $m \ge 1$ відомі. Вибираємо початкову точку $x_0 \in R^n$, величину $\varepsilon > 0$ та переходимо до наступної ітерації з величиною x_0.

Ітеративний процес. Нехай точка $x_k \in R^n$ була знайдена на *k*-й ітерації. Щоб перейти до наступної (*k*+1)-ї ітерації, виконаємо такі дії.

A1. Обчислимо $f(x_k)$ та $g_f(x_k)$. Якщо $f(x_k) - f^* \le \varepsilon$, тоді STOP ($k^* = k$, $x_\varepsilon^* = x_k$).

A2. Обчислимо наступну точку:

$$x_{k+1} = x_k - h_k \frac{g_f(x_k)}{\left\| g_f(x_k) \right\|}, \quad h_k = \frac{m(f(x_k) - f^*)}{\left\| g_f(x_k) \right\|}.$$

A3. Переходимо до наступної (k+1)-ї ітерації з x_{k+1}.

Теорема 2.1. Послідовність $\{x_k\}_{k=0}^{k^*-1}$, породжена методом **A**, задовольняє таку нерівність:

$$\left\| x_{k+1} - x^* \right\|^2 \le \left\| x_k - x^* \right\|^2 - \frac{m^2 (f(x_k) - f^*)^2}{\left\| g_f(x_k) \right\|^2}, \; k = 0,1,2,... \tag{2.5}$$

Доведення. З **A2** для довільного k $(0 \le k \le k^* - 1)$ маємо

$$\left\| x_{k+1} - x^* \right\|^2 = \left\| x_k - x^* - h_k \frac{g_f(x_k)}{\left\| g_f(x_k) \right\|} \right\|^2 = \left\| x_k - x^* \right\|^2 - 2h_k \frac{(x_k - x^*, g_f(x_k))}{\left\| g_f(x_k) \right\|} + h_k^2.$$

Зважаючи на те, що з (2.4) випливає

$$\frac{(x_k - x^*, g_f(x_k))}{\left\| g_f(x_k) \right\|} \ge \frac{m(f(x_k) - f)^*}{\left\| g_f(x_k) \right\|} = h_k,$$

маємо

$$\left\| x_{k+1} - x^* \right\|^2 \le \left\| x_k - x^* \right\|^2 - h_k^2 = \left\| x_k - x^* \right\|^2 - \left(\frac{m(f(x_k) - f)^*}{\left\| g_f(x_k) \right\|} \right)^2,$$

звідки випливають нерівності (2.5). Теорему доведено.

Теорема 2.1 гарантує, що в субградієнтному методі Поляка відстань до точки мінімуму монотонно зменшується. Крім того, задовольняються такі нерівності:

$$(x^* - x_{k+1}, -g_f(x_k)) \ge 0, \; k = 0,1,... \tag{2.6}$$

які означають, що кут між антисубградієнтом у точці x_k і напрямком від точки x_{k-1} до точки мінімуму не тупий. Дійсно, нерівності (2.6) випливають з того, що, використовуючи (2.4), маємо

$$(x^* - x_{k+1}, -g_f(x_k)) = (x_{k+1} - x^*, g_f(x_k)) = (x_k - x^* - h_k \frac{g_f(x_k)}{\|g_f(x_k)\|}, g_f(x_k)) =$$

$$= (x_k - x^*, g_f(x_k) - h_k \|g_f(x_k)\|) = (x_k - x^*, g_f(x_k)) - m(f(x_k) - f^*) \geq 0.$$

Нерівності (2.6) означають, що для опуклої функції $f(x)$, яка задовольняє умову (2.4), h_k визначає величину максимального переміщення у напрямку нормалізованого антисубградієнта, при якому гарантується, що кут між антисубградієнтом і напрямком від точки x_{k-1} до точки мінімуму не буде тупим.

Octave-функція PolyakA. Octave-програма PolyakA знаходить наближення x_ε^* точки мінімуму опуклої функції $f(x)$ від n змінних, що для методу **A** визначається такими вхідними даними: початкова точка x_0; f^* — значення функції в точці мінімуму; $m \geq 1$ — параметр, що визначає величину переміщення у напрямку антисубградієнта, тобто виконання умови (2.4), параметри зупинки ε_f та maxitn.

Програма використовує octave-функцію **function [f, g] = calcfg (x)**, яка обчислює значення функції $f = f(x)$ та її субградієнт у точці x, $g_f(x) \in \partial f(x)$. Назва функції **calcfg** (x) може бути довільною, яку дозволяє синтаксис мови octave. Програма **PolyakA** використовує такі вхідні та вихідні параметри:

```
# Input parameters:
# calcfg - reference to function for calculation of f(x) and g(x)
# x0 - the starting point, x0(1:n)
# fstar - value of the function at the minimum point
# m - length of shift along anti-subgradient (m>=1)
# epsf, maxitn - stop parameters
# intp - print information every intp iteration
# Output parameters:
# x - the minimum point, which was found by the program, x(1:n)
# f - the value of the function f at the point x
# itn - the number of iterations used by the program
# info - exit code (0 = epsf, 4 = maxitn).
```

Статус точки x визначається значенням коду повернення **info** на ітерації itn = k**: info = 0** — зупинка, якщо знайдена точка x_k, для якої $f(x_k) - f^* \le \varepsilon_f$; **info = 4** — зупинка, якщо itn > maxitn (ітераційний процес перевищив максимальну кількість ітерацій).

Програма **PolyakA** є octave-функцією, її код подано нижче.

```
function [x,f,itn,info] = PolyakA(calcfg,x0,fstar,m,          #row01
                                  epsf,maxitn,intp);          #.....
itn=0; x=x0; [f,g] = calcfg(x); dg=norm(g);                   #row02
if(intp>0)                                                    #row03
   printf("itn %4d f %14.6e \n", itn, f); # xprint = x',      #.....
endif                                                         #.....
for(itn = 1:maxitn)                                           #row04
   if(f-fstar < epsf) info = 0; return; endif                 #row05
   g1=g/dg; hs=m*(f-fstar)/dg;                                #row06
   x -= hs * g1;                                              #row07
   [f,g] = calcfg(x); dg=norm(g);                             #row08
   if(mod(itn,intp)==0)                                       #row09
      printf("itn %4d f %14.6e \n",itn,f); # xprint = x',     #.....
   endif                                                      #.....
endfor                                                        #row10
info = 4;                                                     #row11
endfunction                                                   #row12
```

Код програми складається з **14** рядків і більшість із них містить декілька octave-операторів. Ітеративний процес виконується у циклі **for** (рядки 4–10), де для k-ї ітерації субградієнт $g_f(x_k)$ зберігається як вектор-стовпчик ***g***, а нормований субградієнт $g_f(x_k)$ — як вектор-стовпчик ***g*1**. У циклі **for** зупинка реалізується тоді, коли відхилення функції від оптимального значення стає меншим, ніж ε_f (рядок 5).

Використовуючи програму **PolyakA**, можна знайти прийнятно-точні наближення до точки мінімуму гладкої опуклої функції, поверхні рівня якої характеризуються малим ступенем яружності (витягнутості поверхонь рівня

функції). Метод **A** збігається швидко до точки мінімуму опуклої функції, якщо кути між послідовними субградієнтами на кожній ітерації будуть гострими або тупими, близькими до прямого кута.

Недоліком методу **A** є його повільна збіжність для яружних функцій. Нижче буде детальніше розглянута збіжність методу **A** для функцій $f_1(x_1,x_2)=|x_1|+t|x_2|,\ t>1$ та $f_2(x_1,x_2)=x_1^2+tx_2^2,\ t>>1$.

Обчислювальні експерименти для яружних функцій. Нижче описуються результати обчислювальних експериментів з використанням програми **PolyakA**. Експерименти пов'язані з вивченням швидкості збіжності методу **A** для яружних функцій двох змінних — квадратичної функції $f_2(x_1,x_2)=x_1^2+tx_2^2,\ t>>1,\ f^*=0,\ x^*=(0,0)^T$ (задача **quad**) та кусково-лінійної функції $f_1(x_1,x_2)=|x_1|+t|x_2|,\ t>1,\ f^*=0,\ x^*=(0,0)^T$ (задача **sabs**).

Експерименти були проведені на комп'ютері Pentium 2.5 GHz з операційною системою Windows XP/32 з використанням GNU Octave версії 3.0.0. Для задач **quad** і **sabs** використовуються octave-функції для обчислення $f(x)$ та $g_f(x)$ у такому вигляді:

```
function [f,g] = squad(x)
global t;
f=x(1,1)*x(1,1)+t*x(2,1)*x(2,1);
g(1,1) = 2*x(1,1);
g(2,1) = 2*t*x(2,1);
endfunction
```

```
function [f,g] = sabs(x)
global t;
f=abs(x(1,1))+t*abs(x(2,1));
g(1,1) = sign(x(1,1));
g(2,1) = t*sign(x(2,1));
endfunction
```

У випадку негладкої функції двох змінних $f_1(x_1,x_2)=|x_1|+t|x_2|,\ t>1$ швидкість збіжності методу **A** визначається геометричною прогресією зі знаменником

$$q=\sqrt{1-1/t^2} \tag{2.7}$$

і буде дуже повільною для великих значень t. Наприклад, зигзагоподібна траєкторія методу **A** для знаходження точки мінімуму слабко яружної функції $f_1(x_1,x_2)=|x_1|+5|x_2|$ показана на рис. 2.1.

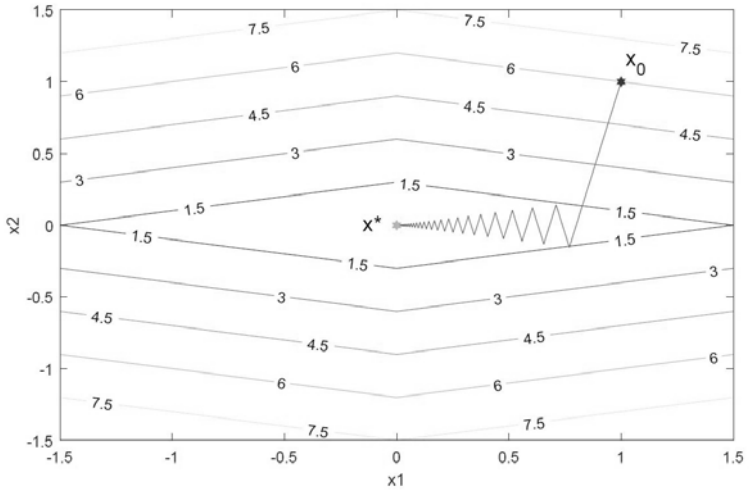


Рис. 2.1. Траєкторія методу **A** для пошуку наближення до точки мінімуму функції

$$f_1(x_1,x_2)=|x_1|+5|x_2|:\ m=1,\ f^*=0,\ x^*=(1,1)^T$$

На рис. 2.2, 2.3 зображені траєкторії методу **A** для мінімізації квадратичної яружної функції $f_2(x_1,x_2)=x_1^2+tx_2^2$. Якщо $t=6$, $m=2$ та $\varepsilon=0.01$, то траєкторія є зигзагоподібною (див. рис. 2.2), при чому для перших п'яти ітерацій метод породжує послідовність $x_0=(1.00,1.00)^T$, $x_1=(0.811,-0.135)^T$, $x_2=(0.338,0.338)^T$, $x_3=(0.274,-0.046)^T$, $x_4=(0.114,0.114)^T$, $x_5=(0.093,-0.015)^T$ (див. рис. 2.2). Якщо ж $t=6$ та $m=1$, то траєкторія, породжена методом **A**, не є зигзагоподібною та прямує до точки мінімуму швидше (див. рис. 2.3).

У табл. 2.1 наведені результати методу **A** для знаходження точки мінімуму функцій $f_1(x_1,x_2)=|x_1|+t|x_2|$ та $f_2(x_1,x_2)=x_1^2+tx_2^2$ для різних ступенів розтягу, які визначаються параметром t. Перший стовпчик містить точності від 10^{-1} до 10^{-10}, з якими обчислюються наближення до точки максимуму. Колонки 2, 3, 4 містять кількості ітерацій, які необхідні для мінімізації кусково-лінійної функції $f_1(x_1,x_2)=|x_1|+t|x_2|$ з $t=100,\ 50,\ 25$ відповідно. Колонки 5, 6, 7 показують

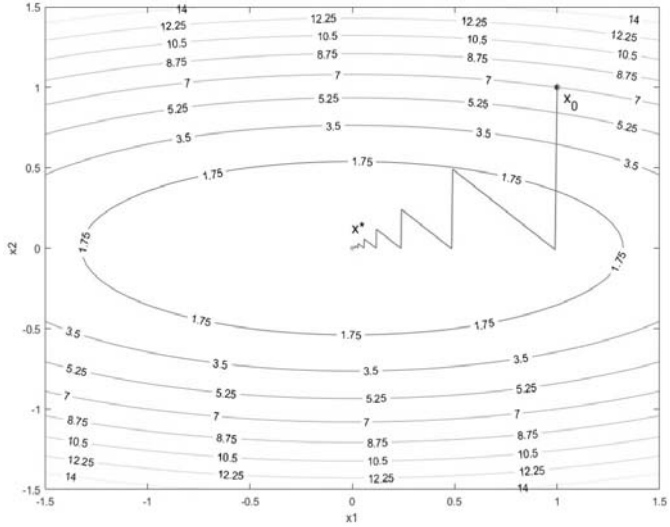


Рис. 2.2. Траєкторія методу **A** пошуку наближення до точки мінімуму квадратичної функції

$$f(x_1, x_2) = x_1^2 + 6x_2^2 : m = 2,\ f^* = 0,\ x_0 = (1,1)^T$$

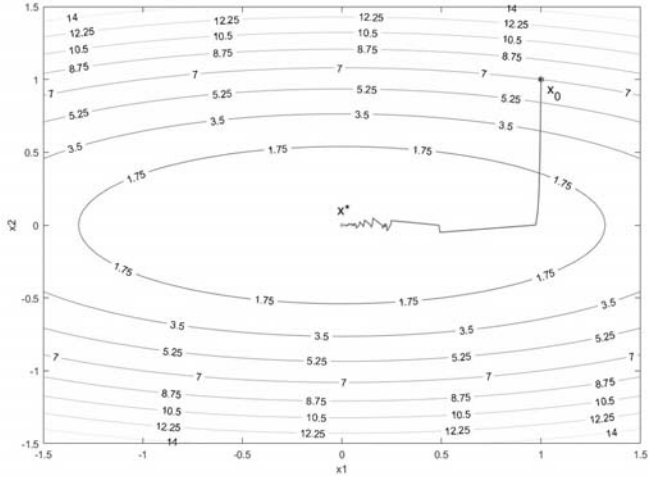


Рис. 2.3. Траєкторія методу **A** пошуку наближення до точки мінімуму квадратичної функції

$$f(x_1, x_2) = x_1^2 + 6x_2^2 : m = 1,\ f^* = 0,\ x_0 = (1,1)^T$$

кількості ітерацій для квадратичної функції $f_2(x_1,x_2)=x_1^2+tx_2^2$ з $t=10000,1000,100$, де кількість ітерацій подається у дужках, якщо параметр $m=1$ використовується для методу **A** замість $m=2$. З колонок 5–7 табл. 2.1 видно, що кількість ітерацій при $m=1$ є значно більшою, ніж кількість ітерацій при $m=2$.

Табл. 2.1. Кількість ітерацій методу **A** для знаходження x_ε^* функцій $f_1(x_1,x_2)=|x_1|+t|x_2|$ та $f_2(x_1,x_2)=x_1^2+tx_2^2: f^*=0,\ x^*=(1,1)^T$

ε_f	$f_1(x_1,x_2)=\|x_1\|+t\|x_2\|$			$f_2(x_1,x_2)=x_1^2+tx_2^2$		
	$t=100$	$t=50$	$t=25$	$t=10000$	$t=1000$	$t=100$
1.0e−01	14930	3721	925	6(139)	6(97)	6(9)
1.0e−02	26443	6600	1645	10(295)	10(127)	10(29)
1.0e−03	37956	9478	2365	12(551)	12(194)	12(51)
1.0e−04	49469	12356	3084	16(646)	16(230)	16(68)
1.0e−05	60982	15234	3804	20(885)	20(324)	20(86)
1.0e−06	72495	18113	4523	22(1003)	22(390)	22(98)
1.0e−07	84008	20991	5243	26(1135)	26(418)	26(121)
1.0e−08	95521	2386	5962	30(1332)	30(485)	28(135)
1.0e−09	107034	26747	6682	32(1518)	32(513)	32(150)
1.0e−10	118547	29625	7401	36(1947)	36(543)	36(169)

З табл. 2.1 бачимо повільну збіжність методу **A**, якщо функції $f_1(x_1,x_2)$ та $f_2(x_1,x_2)$ є яружними. Чим більша витягнутість поверхонь рівня, якій відповідає більше значення параметра t, тим повільніша збіжність методу **A**. Зі збільшенням точності знаходження наближення до точки мінімуму значно зростає й кількість ітерацій. До того ж, ситуація буде гіршою для кусково-лінійної функції $f_1(x_1,x_2)$. Так, наприклад, для досягнення точності 10^{-10} при $t=100$ необхідно виконати 118 547 ітерацій.

Однак скоротивши ступінь розтягу поверхонь рівня у 4 рази, для досягнення тієї ж точності, необхідно суттєво менше ітерацій — 7401. У разі мінімізації квадратичної функції ситуація дещо інша: для $m=2$ кількість

ітерацій практично не залежить від ступеню розтягу. Наприклад, для досягнення тієї ж точності 10^{-10} необхідно лише 36 ітерацій. Якщо використовується параметр $m=1$ для квадратичної функції, то кількість ітерацій зростає при збільшенні параметра t.

Схожою буде ситуація для опуклих функцій $f(x_1,x_2)=x_1^4+10000x_2^4$ та $f(x_1,x_2)=(x_1+1.001x_2)^4+(1.001x_1+x_2)^4$, для яких $m=4$. Так, при точності $\varepsilon_f=10^{-20}$ для знаходження точки мінімуму першої функції знадобилось 4 ітерації при $m=4$ та 50 ітерацій при $m=1$, а для другої функції — 2 ітерації при $m=4$ та 45 ітерацій при $m=1$.

Нижче розглядається модифікація субградієнтного методу з кроком Поляка, де прискорення збіжності порівняно з методом **А** може бути забезпечене вибором матриці перетворення простору.

2.4. Субградієнтний метод з кроком Поляка в перетвореному просторі

Крок Поляка в перетвореному просторі змінних. Розглянемо субградієнтний крок Поляка в перетвореному просторі змінних відповідно до ідеї Шора [7] для прискорення методів, використовуючи лінійні перетворення простору. Ця ідея дала метод еліпсоїдів і *r*-алгоритми (субградієнтні методи з розтягом простору в напрямку послідовних субградієнтів) для опуклої оптимізації [8, 9, 10].

Зробимо заміну змінних $x=By$, де B є невиродженою $n\times n$-матрицею (тобто для неї існує обернена матриця $A=B^{-1}$). Субградієнт $g_\varphi(x)$ опуклої функції $\varphi(y)=f(By)$ в точці $y=Ax$ задовольняє таку умову:

$$(y-y^*, g_\varphi(y))\ge m(\varphi(x)-\varphi^*),\ \forall y\in R^n, \tag{2.8}$$

де $g_\varphi(y) = B^T g_f(x), \varphi^* = \varphi(y^*) = y^* = Ax^*$. Дійсно, оскільки $A = B^{-1}$ і $x = By$, нерівність (2.4) можна переписати у вигляді

$$(A(x - x^*), B^T g_f(x)) \geq m(f(By) - f(By^*)), \forall By \in R^n,$$

звідки отримуємо нерівність (2.8).

Субградієнтний метод з кроком Поляка в перетвореному просторі (визначений невиродженою матрицею B) має такий вигляд:

$$x_{k+1} = x_k - h_k B \frac{B^T g_f(x_k)}{\left\| B^T g_f(x_k) \right\|}, \quad h_k = \frac{m(f(x_k) - f^*)}{\left\| B^T g_f(x_k) \right\|}, \quad k = 0,1,2... \tag{2.9}$$

Тут h_k є кроком Поляка (Агмона – Моцкіна – Шонберга), але в перетвореному просторі змінних $y = Ax$. Це випливає з того, що в перетвореному просторі змінних метод (2.9) записується як субградієнтний процес

$$y_{k+1} = y_k - h_k \frac{g_\varphi(y_k)}{\left\| g_\varphi(y_k) \right\|}, \quad h_k = \frac{m(\varphi(y_k) - \varphi^*)}{\left\| g_\varphi(y_k) \right\|}, \quad k = 0,1,2... \tag{2.10}$$

Крок Поляка в перетвореному просторі змінних має ті самі властивості, що й крок Поляка в початковому просторі. Вони визначаються мінімальним значенням функції φ^* та нерівністю (2.8).

Опис методу В. Щоб знайти точку $x_\varepsilon^* \in R^n$, для якої $f(x_\varepsilon^*) \leq f^* + \varepsilon$, субградієнтний метод з кроком Поляка в перетвореному просторі описується такою ітеративною процедурою.

Ініціалізація. Маємо f^* та $m \geq 1$. Виберемо початкову точку $x_0 \in R^n$, невироджену $n \times n$-матрицю B і величину $\varepsilon > 0$. Переходимо до наступної ітерації з величиною x_0.

Ітеративний процес. Нехай точка $x_k \in R^n$ була знайдена на *k*-й ітерації. Щоб перейти до наступної (*k*+1)-ї ітерації, виконаємо такі дії.

B1. Обчислимо $f(x_k)$ та $g_f(x_k)$. Якщо $f(x_k) - f^* \le \varepsilon$, тоді STOP ($k^* = k$, $x_\varepsilon^* = x_k$).

B2. Обчислимо наступну точку:

$$x_{k+1} = x_k - h_k B \frac{B^T g_f(x_k)}{\left\| B^T g_f(x_k) \right\|}, \quad h_k = \frac{m(f(x_k) - f^*)}{\left\| B^T g_f(x_k) \right\|}.$$

B3. Переходимо до наступної (k+1)-ї ітерації з x_{k+1}.

Теорема 2.2. Послідовність $\{x_k\}_{k=0}^{k^*-1}$, породжена методом **B**, задовольняє нерівності

$$\left\| A(x_{k+1} - x^*) \right\|^2 \le \left\| A(x_k - x^*) \right\|^2 - \frac{m^2 (f(x_k) - f^*)^2}{\left\| B^T g_f(x_k) \right\|^2}, \quad k = 0,1,2,... \tag{2.11}$$

Доведення. З **B2** для довільного k $(0 \le k \le k^* - 1)$ маємо

$$\left\| A(x_{k+1} - x^*) \right\|^2 = \left\| A(x_k - x^*) - h_k \frac{B^T g_f(x_k)}{\left\| B^T g_f(x_k) \right\|} \right\|^2 =$$

$$= \left\| A(x_k - x^*) \right\|^2 - 2h_k \frac{(x_k - x^*, g_f(x_k))}{\left\| B^T g_f(x_k) \right\|} + h_k^2.$$

Беручи до уваги, що з (2.4) випливає нерівність

$$\frac{(x_k - x^*, g_f(x_k))}{\left\| B^T g_f(x_k) \right\|} \ge \frac{m(f(x_k) - f)^*}{\left\| B^T g_f(x_k) \right\|} = h_k,$$

маємо

$$\left\| A(x_{k+1} - x^*) \right\|^2 \le \left\| A(x_k - x^*) \right\|^2 - h_k^2 = \left\| A(x_k - x^*) \right\|^2 - \left(\frac{m(f(x_k) - f)^*}{\left\| B^T g_f(x_k) \right\|} \right)^2,$$

що доводить нерівність (2.11). Теорема доведена.

Теорема 2.2 гарантує, що в субградієнтному методі з кроком Поляка в перетвореному просторі змінних відстань до точки мінімуму зменшується монотонно в перетвореному просторі змінних. Крім того, задовольняються такі нерівності:

$$(A(x^* - x_{k+1}), -B^T g_f(x_k)) \geq 0,\ k = 0,1,... \tag{2.12}$$

які можуть бути переписані як нерівності

$$(y^* - y_{k+1}, -g_\varphi(y_k)) \geq 0,\ k = 0,1,... \tag{2.13}$$

Дійсно, нерівності (2.12) випливають з того, що, використовуючи (2.8) і (2.10), маємо:

$$(x^* - x_{k+1}, -g_f(x_k)) = (x_{k+1} - x^*, g_f(x_k)) = (x_k - x^* - h_k B \frac{B^T g_f(x_k)}{\left\| B^T g_f(x_k) \right\|}, g_f(x_k)) =$$

$$= (x_k - x^*, g_f(x_k) - h_k \left\| B^T g_f(x_k) \right\|) = (x_k - x^*, g_f(x_k)) - m(f(x_k) - f^*) \geq 0.$$

Нерівності (2.13) означають, що для опуклої функції $\varphi(x)$, яка задовольняє умову (2.8), крок h_k визначає величину максимального переміщення у напрямку нормалізованого антисубградієнта, при якому гарантується, що кут між антисубградієнтом у точці y_k і напрямком від точки y_{k+1} до точки мінімуму y^* не буде тупим у перетвореному просторі змінних.

Octave-функція PolyakB і обчислювальні експерименти. Octave-функція **PolyakB** знаходить наближення x_ε^* опуклої функції $f(x)$, яка для методу **B** визначається такими вхідними даними: B — $n \times n$-матриця перетворення простору; x_0 — початкова точка; f^* — значення функції в точці мінімуму; $m \geq 1$ — параметр, що задовольняє умову (2.4); параметри зупинки ε_f та maxitn. Її код наведено нижче.

```
# Input parameters:
# calcfg - name of the function for calculation of f(x) and g(x)
# B - n*n-matrix for transformation of space
# x0 - the starting point, x0(1:n)
# fstar - value of the function at the minimum point
# m - length of shift along anti-subgradient (m>=1)
# epsf, maxitn - stop parameters
# intp - print information every intp iteration
# Output parameters:
# x - the minimum point, which was found by the program, x(1:n)
# f - the value of the function f at the point x
```

```
# itn - the number of iterations used by the program
# info - exit code (0 = epsf, 4 = maxitn)

function [x,f,itn,info] = PolyakB(calcfg,B,x0,fstar,m,    #row01
epsf,maxitn,intp);                                        #.....
itn=0; x=x0; [f,g] = calcfg(x);                           #row02
if(intp>0)                                                #row03
printf("itn %4d f %14.6e \n", itn, f); # xprint = x',     #.....
endif                                                     #.....
for((itn = 1:maxitn)                                      #row04
   if(f-fstar < epsf) info = 0; return; endif             #row05
g1=B'*g; dg1=norm(g1);                                    #row06
g2=g1/dg1; hs=m*(f-fstar)/dg1;                            #row07
x -= hs * B * g2;                                         #row08
[f,g] = calcfg(x); dg=norm(g);                            #row09
if(mod(itn,intp)==0)                                      #row10
printf("itn %4d f %14.6e \n",itn,f); # xprint = x',       #.....
endif                                                     #.....
endfor                                                    #row11
info = 4;                                                 #row12
endfunction                                               #row13
```

Використовуючи програму **PolyakB**, можна знайти досить точні наближення до точки мінімуму опуклої функції. Якщо матрицю B вибрати так, що в перетвореному просторі змінних поверхні рівня яружних функцій витягнуті менше, ніж у вихідному просторі змінних, то метод **B** збігатиметься швидше, ніж метод **A**.

Наприклад, якщо матриця $B = diag\left(1;0.5\right)$, то для функції $f_1(x_1,x_2) = |x_1| + t|x_2|, t > 1$ швидкість збіжності методу **B** визначається геометричною прогресією із знаменником $q = \sqrt{1-4/t^2}$, який менший, ніж $q = \sqrt{1-1/t^2}$ для методу **A** (див. (2.7)). Якщо $m = 2, B = diag\left(1;0.7\right)$ і $\varepsilon = 0.01$, тоді при мінімізації функції $f_2(x_1,x_2) = x_1^2 + 6x_2^2$ за перших п'ять ітерацій метод **B** генерує послідовність $x_0 = (1.00,1.00)^T$, $x_1 = (0.624,-0.104)^T$, $x_2 = (0.136,0.136)^T$, $x_3 = (0.085,-0.014)^T$.

Для яружних функцій субградієнтний метод Поляка з перетворенням простору (метод **B**) буде ефективнішим, ніж субградієнтний метод Поляка без

перетворення простору (метод **A**). Це узгоджується з кількістю ітерацій методу **B** для знаходження десяти послідовно уточнених наближень точки мінімуму функції $f_1(x_1,x_2)=|x_1|+10|x_2|$ для шести різних матриць B, кожній з яких відповідає стовпчик у табл. 2.2.

Матриці B одержано як результат розтягу простору змінних у напрямку x_2 з коефіцієнтами розтягу $\alpha=1;1.5;2;3;4;5$. Матриця B має вигляд $B=\begin{pmatrix}1 & 0\\ 0 & \frac{1}{\alpha}\end{pmatrix}$, і, якщо $\alpha=1$, вона збігається з одиничною матрицею. Випадок $\alpha=1$ відповідає методу **A**.

Табл. 2.2. Кількість ітерацій методу **B** для мінімізації функції $f_1(x_1,x_2)=|x_1|+10|x_2|$, $x_0=(1,1)^T$

ε_f	$\alpha=1$	$\alpha=1.5$	$\alpha=2$	$\alpha=3$	$\alpha=4$	$\alpha=5$
1.0e−01	147	63	33	6	10	9
1.0e−02	262	114	62	19	17	13
1.0e−03	377	165	91	31	24	18
1.0e−04	492	216	119	44	31	22
1. 0e−05	607	268	148	57	38	27
1.0e−06	722	319	177	70	45	31
1.0e−07	837	370	206	82	53	36
1.0e−08	952	421	234	95	60	40
1.0e−09	1068	472	263	108	67	45
1.0e−10	1183	523	292	121	74	49

З табл. 2.2 бачимо, що кількість ітерацій методу Поляка з перетворенням простору зменшується монотонно, коли ступінь яружної функції $\varphi_1(y_1,y_2)=|y_1|+\frac{10}{\alpha}|y_2|$ зменшується у перетвореному просторі змінних (що відповідає збільшенню коефіцієнта розтягу α).

Ситуація зі швидкістю збіжності буде складнішою для суттєво-яружної функції $f_3(x_1,x_2)=\max\left\{x_1^2+(2x_2-2)^2-3,\ x_1^2+(x_2+1)^2\right\}$. З табл. 2.3 бачимо, що монотонного зменшення кількості ітерацій не спостерігається, якщо величина α, коефіцієнт розтягу простору в напрямку x_2, збільшується. Але є певний

розрив, який має місце при значенні коефіцієнта $\alpha = 2$ (відповідає колонці з $\alpha = 2$).

Табл. 2.3. Кількість ітерацій методу **B** для мінімізації функції

$f_3(x_1, x_2) = \max\left\{x_1^2 + (2x_2 - 2)^2 - 3,\ x_1^2 + (x_2 + 1)^2\right\}$, $x_0 = (1,1)^T$, maxitn = 100 000

ε_f	$\alpha = 1$	$\alpha = 1.5$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
1.0e−01	16	4	4	5	7	8
1.0e−02	162	37	4	6	7	9
1.0e−03	1604	679	5	6	8	9
1.0e−04	16004	7079	5	6	9	46
1. 0e−05	–	71079	6	8	8061	6206
1.0e−06	–	–	6	–	98061	63806
1.0e−07	–	–	6	–	–	–

На рис. 2.4 зображені траєкторії методів **A** ($\alpha = 1$) та **B** ($\alpha = 2$) суцільною та штриховою лініями відповідно. На траєкторії методу **B** зображені всі точки, отримані алгоритмом, на траєкторії методу **A** зображені лише перші чотири точки.

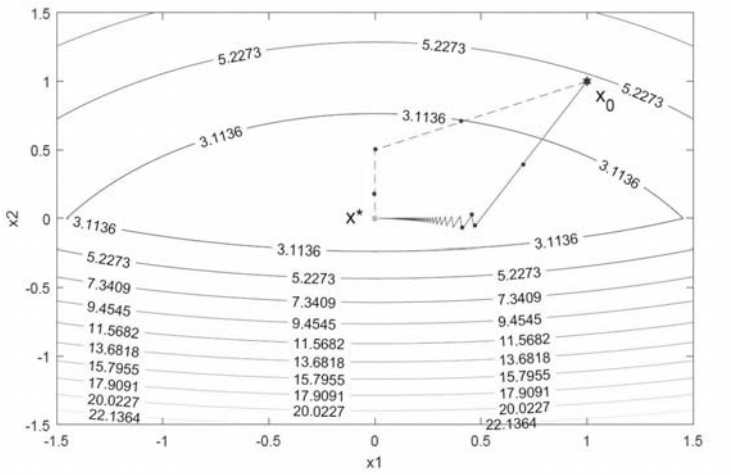


Рис. 2.4. Траєкторія методу **A** (суцільна лінія) і траєкторія методу **B** (штрихова лінія) для знаходження наближення до точки мінімуму кусочно-квадратичної функції

$f_3(x_1, x_2)$: $m = 1$, $f^* = 1$, $x_0 = (1,1)^T$, B = diag(1;0.5), $x^* = (0,0)$

Розглянемо використання програм **PolyakA** і **PolyakB** для мінімізації опуклої квадратичної функції $f_4(x) = \|Ax - b\|^2$ від змінних $x \in R^n$, де $A = \|a_{ij}\|_{i,j=1}^{l,n}$ — довільна $l \times n$-матриця і l-вимірний вектор b такий, що його компоненти $b_i = \sum_{j=1}^{n} a_{ij}$, $i = 1,..,l$. Для функції $f^* = 0$ і, якщо матриця A має повний ранг, то функція має єдину точку мінімуму $x^* = (1,1,..,1)^T$.

Octave-функція для обчислення значення функції $f_4(x)$ та її градієнта має такий вигляд:

```
function [f,g] = fgfun4(x)
global A b;
temp=A*x- b;
f = temp'*temp;
g=2*A'*temp;
endfunction
```

Розглянемо тестовий приклад з 500×100-матрицею, що має вигляд

$$A = \begin{pmatrix} 100 & 0 & 0...0 \\ 0 & 100 & 0...0 \\ & A_1 & \end{pmatrix},$$

де A_1 — 498×100-матриця, яка отримана за допомогою генератора випадкових чисел з відрізка [0, 3].

Код octave-програми для порівняння методів **A** і **B**, де матриця $B = diag(0.1; 0.1; 1;...; 1)$, є таким:

```
global A b;
n=100; x0 = zeros(n,1); B=diag([0.1 0.1 ones(1,n-2)]);
rand("seed", 2018); A1=3*rand(498,n);
A=[diag([100 100]) zeros(2,n-2); A1]; b=sum(A')';
m=2; fstar=0.d0; epsf = 0.0001; maxitn = 50000; intp=1000;
for(itn = 1:6)
 [xA,fA,itnA,infoA]=PolyakA(@fgfun4,x0,fstar,m,epsf,maxitn,intp);
 [xB,fB,itnB,infoB]=PolyakB(@fgfun4,B,x0,fstar,m,epsf,maxitn,intp);
 epsf, itnA, dnA=norm(xA-ones(n,1)), itnB, dnB=norm(xB-ones(n,1)),
 epsf=epsf/100;
endfor
```

Результати роботи цієї програми є такими:

```
epsf=1.0e-04  itnA= 695  dnA=8.0612e-04  itnB= 86  dnB=1.2175e-04
epsf=1.0e-06  itnA=1085  dnA=8.2861e-05  itnB=108  dnB=1.1773e-05
epsf=1.0e-08  itnA=1491  dnA=8.3470e-06  itnB=130  dnB=1.1353e-06
epsf=1.0e-10  itnA=1901  dnA=8.3881e-07  itnB=150  dnB=1.3526e-07
epsf=1.0e-12  itnA=2313  dnA=8.3994e-08  itnB=172  dnB=1.3018e-08
epsf=1.0e-14  itnA=2725  dnA=8.4440e-09  itnB=194  dnB=1.2523e-09
```

З цієї таблиці видно, що метод **B** потребує значно меншу кількість ітерацій, ніж метод **A**. Наприклад, щоб досягнути точності 10^{-14}, методу **A** необхідно виконати 2725 ітерацій, тоді як метод **B** потребує лише 194 ітерації.

2.5. Метод та програма amsg2p

Субградієнтний метод **amsg2p** дозволяє знайти точку мінімуму опуклої функції $f(x)$ при відомому значенні f^* або його верхній оцінці [10, с. 386–387]. Його детальний опис наведено у звіті [11]. У назві методу «ams» означає, що в ітераційному процесі використовується крок Агмона – Моцкіна – Шонберга (крок Поляка) в напрямку нормованого антисубградієнта, а «g2p» вказує, що ams-крок використовується у просторі змінних, перетвореному за допомогою двох послідовних субградієнтів (g2) та агрегатного вектора (p). Перетворення простору в методі **amsg2p** проводиться за допомогою одно-рангового еліпсоїдального оператора [12, 13] і лише на тих ітераціях методу, коли тупим є хоча б один із кутів — або кут між двома послідовними субградієнтами, або кут між останнім субградієнтом та агрегатним вектором, який є опуклою комбінацією обчислених на попередніх ітераціях субградієнтів.

Нехай $f(x)$ — опукла функція векторного аргументу $x \in R^n$, а її субградієнт $g_f(x)$ задовольняє умову

$$(x - x^*, g_f(x)) \geq \gamma(f(x) - f^*), \quad \forall x \in R^n, \forall x^* \in X^*, \quad \gamma \geq 1. \qquad (2.14)$$

Тут X^* — множина точок мінімуму функції $f(x)$; f^* — мінімальне значення функції $f(x)$: $f^* = f(x^*)$, $x^* \in X^*$. Параметр γ вважається відомим та

введений для врахування спеціальних класів опуклих функцій. Він позначається інакше, ніж аналогічний йому параметр m у співвідношенні (2.4), щоб відрізнити випадок $X^* = x^*$ (точка мінімуму єдина) від загального випадку. Значення $\gamma = 1$ забезпечує виконання умови (2.14) для довільної опуклої функції.

Метод **amsg2p** дає можливість або знайти таку точку, де значення опуклої функції $f(x)$ менше або дорівнює $f_{min} + \varepsilon_f$, або гарантує достатню умову того, що точки, в якій значення $f(x)$ дорівнює f_{min}, у кулі заданого радіуса не існує. У першому випадку метод **amsg2p** знаходить точку $x_\varepsilon^* \in \{x : f(x) - f_{min} \le \varepsilon_f\}$ і відповідний їй номер ітерації позначається k_ε^*, а в другому — зупиняється з повідомленням «точки не існує».

Метод **amsg2p** полягає в наступному.

На ітерації $k = 0$ задані: початкове наближення $x_0 \in R^n$; початковий радіус r_0 такий, що $\|x_0 - x^*\| \le r_0$; досить мале $\varepsilon_f > 0$. Обчислимо $f(x_0)$ та $g_f(x_0)$. Якщо $f(x_0) - f_{min} \le \varepsilon$, тоді $x_\varepsilon^* = x_0$, $k_\varepsilon^* = 0$, і алгоритм закінчує роботу. Інакше, визначаємо $h_0 = \dfrac{\gamma(f(x_0) - f_{min})}{\|g_f(x_0)\|}$, $\xi_0 = \dfrac{g_f(x_0)}{\|g_f(x_0)\|} \in R^n$, $p_0 = 0 \in R^n$, $B_0 = I$ — одинична $n \times n$ −матриця. Перейдемо до наступної ітерації.

Нехай на k-й ітерації отримано $x_k \in R^n$, h_k, r_k, $\xi_k \in R^n$, $p_k \in R^n$, B_k — матриця $n \times n$. Для переходу на $(k+1)$-у ітерацію виконаємо такі кроки.

Крок 1. Обчислимо $t_k = h_k / r_k$. Якщо $t_k > 1$, тоді «точки не існує» і алгоритм завершує роботу. Інакше — визначимо $r_{k+1} = r_k \sqrt{1 - t_k^2}$ та обчислимо чергове наближення

$$x_{k+1} = x_k - h_k B_k \xi_k.$$

Крок 2. Обчислимо $f(x_{k+1})$ і $g_f(x_{k+1})$. Якщо $f(x_{k+1}) - f_{min} \le \varepsilon$, то $x_\varepsilon^* = x_{k+1}$, $k_\varepsilon^* = k+1$ і алгоритм завершує роботу. Інакше — визначимо

$$\xi_{k+1} = \frac{B_k^T g_f(x_{k+1})}{\left\| B_k^T g_f(x_{k+1}) \right\|}, \quad h_{k+1} = \frac{\gamma(f(x_{k+1}) - f_{min})}{\left\| B_k^T g_f(x_{k+1}) \right\|}.$$

Крок 3. Обчислимо $\lambda_1 = -p_k^T \xi_{k+1}$ і $\lambda_2 = -\xi_k^T \xi_{k+1}$. Визначимо

$$p_{k+1} = \begin{cases} \dfrac{\lambda_1}{\sqrt{\lambda_1^2 + \lambda_2^2}} p_k + \dfrac{\lambda_2}{\sqrt{\lambda_1^2 + \lambda_2^2}} \xi_k, & \text{якщо } \lambda_1 > 0 \; i \; \lambda_2 > 0, \\ p_k, & \text{якщо } \lambda_1 > 0 \; i \; \lambda_2 \le 0, \\ \xi_k, & \text{якщо } \lambda_1 \le 0 \; i \; \lambda_2 > 0, \\ 0, & \text{якщо } \lambda_1 \le 0 \; i \; \lambda_2 \le 0. \end{cases}$$

Крок 4. Обчислимо $\mu_k = p_{k+1}^T \xi_{k+1}$. Якщо $-1 < \mu_k < 0$, то обчислимо

$$B_{k+1} = B_k + (B_k \eta) \xi_{k+1}^T, \text{ де } \eta = \left(\frac{1}{\sqrt{1 - \mu_k^2}} - 1 \right) \xi_{k+1} - \frac{\mu_k}{\sqrt{1 - \mu_k^2}} p_{k+1}$$

і перерахуємо

$$h_{k+1} = \frac{h_{k+1}}{\sqrt{1 - \mu_k^2}}, \qquad p_{k+1} = \frac{1}{\sqrt{1 - \mu_k^2}} (p_{k+1} - \mu_k \xi_{k+1}).$$

Інакше — визначимо $B_{k+1} = B_k$ і $p_{k+1} = 0$.

Крок 5. Перейдемо до наступної ітерації з x_{k+1}, h_{k+1}, r_{k+1}, ξ_{k+1}, p_{k+1}, B_{k+1}.

Теорема 2.3 [11]. Нехай $A_k = B_k^{-1}$, $A_{k+1} = B_{k+1}^{-1}$. Якщо $f_{min} \ge f^*$ та $X^* = x^*$, то для всіх точок, згенерованих методом **amsg2p**, справедливі нерівності

$$\left\| A_{k+1}(x_{k+1} - x^*) \right\|^2 \le \left\| A_k(x_k - x^*) \right\|^2 - \left(\frac{\gamma(f(x_k) - f_{min})}{\left\| B_k^T g_f(x_k) \right\|} \right)^2, \; k = 0,1,\ldots, \qquad (2.15)$$

Теорема 2.3 означає, що в кожному черговому перетвореному просторі змінних відстань до точки мінімуму монотонно зменшується. Завдяки цьому на кожній ітерації $k > 1$ виконується нерівність

$$\left\| A_k(x_k - x^*) \right\|^2 \le \left\| x_0 - x^* \right\|^2 - \sum_{i=0}^{k-1} \left(\frac{\gamma(f(x_i) - f_{min})}{\left\| B_i^T g_f(x_i) \right\|} \right)^2 = r_0^2 - \sum_{i=0}^{k-1} h_i^2 = r_k^2,$$

за допомогою якої забезпечується достатня умова відсутності точки x_ε^* при $f_{min} < f^* - \varepsilon_f$ (реалізовано на кроці 1 методу **amsg2p**).

Подібно до того, як це зроблене в r-алгоритмах Шора, перетворення простору в методі **amsg2p** направлене на зменшення ступеня яружності поверхонь рівня опуклих функцій. Для яружних функцій це забезпечує прискорену збіжність методу при довільній початковій стартовій точці x_0 та досить малих значеннях параметра ε_f ($\varepsilon_f \approx 10^{-14} \div 10^{-10}$).

А якщо функція не яружна, то метод **amsg2p** переходить у відомий субградієнтний метод Поляка:

$$x_{k+1} = x_k - h_k \frac{g_f(x_k)}{\|g_f(x_k)\|}, \quad h_k = \frac{\gamma\left(f(x_k) - f^*\right)}{\|g_f(x_k)\|}, \quad k = 0,1,2,\ldots. \qquad (2.16)$$

який описано в підрозділі 2.3. Тут величину h_k називають кроком Поляка, або кроком Агмона – Моцкіна − Шонберга, які вперше використали таке регулювання кроку в релаксаційному субградієнтному методі для знаходження хоча б одного розв'язку сумісної системи лінійних нерівностей.

Теорема 2.4 [11]. Для усіх точок, згенерованих методом (2.16), справедливі нерівності

$$\|x_{k+1} - x^*\|^2 \le \|x_k - x^*\|^2 - \left(\frac{\gamma(f(x_k) - f^*)}{\|g_f(x_k)\|}\right)^2, \quad \forall x^* \in X^*, k = 0,1,\ldots, \qquad (2.17)$$

Програмна реалізація методу **amsg2p** виконана мовою Octave [14]. Однойменна програма **amsg2p** знаходить точку x_ε^*, де значення опуклої функції $f(x_\varepsilon^*) \le f_{\min} + \varepsilon_f$ і визначається заданими: стартовою точкою x_0; радіусом r_0, параметром γ (зміщення по опуклості), параметрами зупинки ε_g, ε_x та **maxitn**.

Програма використовує octave-функцію **function [f,g] = calcfg(x)**, яка в точці x обчислює значення функції $f = f(x)$ та її субградієнта $g = g_f(x)$. Ім'я функції **calcfg(x)** може бути довільним, яке допускає синтаксис мови Octave.

Програма **amsg2p** використовує такі вхідні та вихідні параметри:

```
%  Вхідні параметри:
%    calcfg – ім'я функції calcfg(x) для обчислення f і g
%    x0  -- стартова точка
%    fmin – мінімальне значення функції (або його оцінка)
%    gamma – параметр зміщення по опуклості
%    rad – радіус початкової кулі з центром у точці x0
%    epsf, epsg, maxitn – параметри зупинки
%  Вихідні параметри:
%    x -- знайдена точка
%    f -- значення функції f в точці x
%    itn -- кількість проведених ітерацій
%    istop -- код зупинки (0=epsf, 1=epsg, 2=maxitn, 3=error).
```

Точка x_{itn} — це остання точка ітераційного процесу та її статус характеризується значенням параметра зупинки **istop** на ітерації **itn**:

1) зупинка відбулася за умовою $f(x_{itn}) \le f_{\min} + \varepsilon_f$ (**istop=0**);
2) зупинка відбулася за умовою $\| g_f(x_{itn}) \| \le \varepsilon_g$ (**istop=1**);
3) зупинка відбулася за умовою **itn>maxitn** (перевищена максимальна кількість ітерацій) (**istop=2**);
4) зупинка відбулася тому, що отримана достатня умова того, що точки, де значення $f(x)$ дорівнює f_{min}, не існує в кулі заданого радіусу (**istop=3**) (ця зупинка вважається аварійною і пов'язана або з тим, що вказане значення **fmin** менше від мінімального значення функції, або початковий радіус **rad** занижений і його варто збільшити).

Програма **amsg2p** оформлена як octave-функція:

```
# octave-function of amsg2p-method
function [x,f,itn,istop]=amsg2p(calcfg,x0,fmin,gamma,rad,      # row001
                                epsf,epsg,maxitn);
itn=0; rad1=rad; B=eye(length(x0)); x=x0;                      # row002
[f,g]=calcfg(x); if(f-fmin<epsf) istop=0; return; endif        # row003
dg=norm(g); if(dg<epsg) istop=1; return; endif                 # row004
hs=gamma*(f-fmin)/dg; g1=g/dg; p=zero=zeros(length(x),1);      # row005
printf("itn %4d f %21.13e rad %10.2e \n",itn, f, rad1);        # row006
for (itn = 1:maxitn)                                           # row007
   tmp=hs/rad1; if(tmp>1.d0) istop=3; return; endif            # row008
   rad1=rad1*sqrt(1.d0-tmp)*sqrt(1.d0+tmp);                    # row009
   x -= hs * B * g1; [f,g] = calcfg(x);                        # row010
   if(f-fmin < epsf) istop=0; return; endif                    # row011
```

```
    if(norm(g) < epsg) istop=1; return; endif                    # row012
    g2=B'*g; dg2=norm(g2); g2=g2/dg2; t1=-p'*g2; t2=-g1'*g2;     # row013
    if(t1>0.d0) if(t2>0.d0) tmp=sqrt(t1*t1+t2*t2);               # row014
      t1=t1/tmp; t2=t2/tmp; p=t1*p+t2*g1; endif                  # row015
    else if(t2>0.d0) p=g1; else p=zero; endif endif              # row016
    hs=gamma*(f-fmin)/dg2; tcos=p'*g2; tmp=1.d0;                 # row017
    if(-1.d0<tcos)&&(tcos<0.d0) tmp=sqrt((1-tcos)*(1+tcos));     # row018
      g1=(1.d0/tmp-1.d0)*g2-(tcos/tmp)*p; B+=B*g1*g2';           # row019
      hs=hs/tmp; p=(p-tcos*g2)/tmp; else p=zero; endif           # row020
    g1=g2;                                                       # row021
    printf("itn %4d f %21.13e rad %9.1e cos %9.1e dB %9.1e\n",   # row022
           itn, f, rad1, tcos,tmp);
  endfor                                                         # row023
  istop=2;                                                       # row024
  endfunction                                                    # row025
```

У табл. 2.4 та 2.5 наведені порівняльні результати обчислювальних експериментів для знаходження за допомогою методів **A**, **B** та **amsg2p** точки мінімуму квадратичної функції $f_4(x) = \|Ax - b\|^2$, для якої обчислення значення функції та її градієнта реалізує octave-функція **`fgfun4(x)`** (див. підрозділ 2.4, с. 52). Розглядався той самий тестовий приклад з 500×100-матрицею $A = \begin{pmatrix} 100 & 0 & 0...0 \\ 0 & 100 & 0...0 \\ & A_1 & \end{pmatrix}$, де A_1 — 498×100-матриця, яка була отримана за допомогою генератора випадкових чисел з інтервалів [0, 3] та [0, 5] відповідно.

Табл. 2.4. Збіжність методів **A**, **B, amsg2p** при мінімізації функції

$$f_4(x) = \|Ax - b\|^2,\ A_1 \in [0,3]^{498\times 100}$$

Epsf	**Метод A**		**Метод B**		**Метод amsg2p**	
	itn	$\|x_{itn} - x^*\|$	**itn**	**time**	**itn**	**time**
1.00E−04	695	8.0612E−04	86	1.2175E−04	12	2.4107E−04
1.00E−06	1085	8.2861E−05	108	1.1773E−05	14	3.6769E−05
1.00E−08	1491	8.3470E−06	130	1.1353E−06	19	2.1710E−06
1.00E−10	1901	8.3881E−07	150	1.3526E−07	22	3.1324E−07
1.00E−12	2313	8.3994E−08	172	1.3018E−08	24	4.2297E−08
1.00E−14	2725	8.4439E−09	194	1.2523E−09	27	3.8820E−09
1.00E−16	3139	8.4141E−10	216	1.2040E−10	30	3.1344E−10
1.00E−18	3553	8.3820E−11	238	1.1637E−11	33	2.6745E−11
1.00E−20	3967	8.4642E−12	260	1.1283E−12	35	5.0263E−12

Табл. 2.4. Збіжність методів **A**, **B, amsg2p** при мінімізації функції

$$f_4(x) = \|Ax - b\|^2, \quad A_1 \in [0,5]^{498\times 100}$$

epsf	**Метод A**		**Метод B**		**Метод amsg2p**	
	itn	$\|x_{itn} - x^*\|$	**itn**	**time**	**itn**	**time**
1.00E−04	265	4.9693E−04	352	1.0379E−04	12	2.1617E−04
1.00E−06	399	5.2130E−05	438	1.0776E−05	15	2.5641E−05
1.00E−08	539	5.2116E−06	526	1.0616E−06	20	1.3339E−06
1.00E−10	681	5.1205E−07	614	1.0459E−07	23	1.9892E−07
1.00E−12	821	5.2309E−08	700	1.0862E−08	26	1.8676 E−08
1.00E−14	963	5.1901E−09	788	1.0704E−09	28	2.4180E−09
1.00E−16	1105	5.1594E−10	876	1.0541E−10	31	1.9506E−10
1.00E−18	1247	5.1462E−11	964	1.0372E−11	33	2.1189E−11
1.00E−20	1389	5.1388E−12	1052	1.0594E−12	36	2.1323E−12

У розрахунках використовувався параметр $\gamma = 2$, $f_{min} = f^* = 0$, $x_0 = (0,\ldots,0)^T$. З табл. 2.4 та 2.5 видно, що робота методу **A** є більш ніж удвічі кращою, коли випадкові числа для заповнення матриці A_1 беруться з інтервалу [0, 5], а не з інтервалу [0, 3]. Натомість метод **B** демонструє прямо протилежні результати і працює більш ніж учетверо швидше, якщо випадкові числа беруться з інтервалу [0, 3]. А от робота методу **amsg2p** не залежить від вибору інтервалу — результати майже повністю збігаються.

В обох прикладах кількість ітерацій методу **amsg2p** є суттєво меншою, ніж кількості ітерацій методів **A** та **B**. Тому метод **amsg2p** можна успішно використовувати для мінімізації яружних опуклих функцій, а матрично-векторні операції роблять його перспективним у системах паралельних та розподілених обчислень.

Наявні бібліотеки стандартних програм для паралельних матрично-векторних операцій дозволяють за короткий термін адаптувати алгоритм для ефективної роботи з використанням векторних процесорів на основі графічних прискорювачів (GPU).

2.6. Висновки

Розглянуті в розділі субградієнтні методи з кроком Поляка можуть бути використані для знаходження допустимої точки сумісної системи опуклих нерівностей $f_i(x) \le 0,\ i = 1,..,l,\ x \in R^n$. Ця задача еквівалентна мінімізації негладкої опуклої функції $\psi(x) = \max\left\{0, \max\limits_{1 \le i \le l} f_i(x)\right\}$, для якої $\psi^* = 0$. Частковим випадком системи опуклих нерівностей є система лінійних нерівностей.

Задачі лінійного програмування можна звести до цього випадку, використовуючи обмеження для прямих і двоїстих задач. При мінімізації $\psi(x)$ рекомендується використовувати значення параметра $m = 1$. Параметр $m = 2$ може бути використаний для знаходження розв'язку сумісної системи лінійних рівнянь (повного рангу, недовизначеної або перевизначеної).

Список літератури

1. Стецюк П. И. Ускорение субградиентного метода Поляка. Теорія оптимальних рішень. 2012. № 11. С. 151−160.
2. Stetsyuk P., Stovba V., Chernousova Z. Subgradient Method with Polyak's Step in Transformed Space. Optimization and Applications, OPTIMA 2018, Communications in Computer and Information Science 974 (2019). P. 49−63.
3. Поляк Б. Т. Минимизация негладких функционалов. Вычислительная математика и математическая физика. 1969. Т. 9. № 3. С. 509−521.
4. Agmon S. The relaxation method for linear inequalities. Canad. J. Math. 6 (1954). P. 382−392.
5. Motzkin T., Schoenberg I. J. The relaxation method for linear inequalities. Canad. J. Math. 6 (1954). P. 393−404.
6. Еремин И. И. Обобщение релаксационного метода Агмона − Моцкина. УМН. 1965. Т. XX. Вып. 2 (122). С. 183−187.

7. Сергиенко И. В., Стецюк П. И. О трех научных идеях Н. З. Шора. Кибернетика и системный анализ. 2012. № 1. С. 4−22.
8. Шор Н. З. Методы минимизации недифференцируемых функций и их приложения. Киев: Наукова думка, 1979. 200 с. (English transl.: Shor N. Z. Minimization Methods for Non-Differetiable Functions. Berlin: Springer-Verlag, 1985. 178 p.)
9. Shor N. Z. Nondifferentiable optimization and polynomial problems. Boston; Dordrecht; London: Kluwer Academic Publishers. 1998. 412 p.
10. Стецюк П. И. Методы эллипсоидов и r-алгоритмы. Кишинэу: Эврика, 2014. 488 с.
11. Стецюк П. І., Журбенко М. Г., Березовський О. А. та ін. Розробити субградієнтні алгоритми розв'язання задач оптимізації з гарантованою точністю. Заключний звіт про науково-дослідну роботу № держ. реєстрації 0114U001055. Київ: Ін-т кібернетики імені В. М. Глушкова НАН України, 2016. 235 с.
12. Stetsyuk P. I. 2d-Ellipsoid of optimal volume and its applications. In: L. N. Polyakova (ed.) Constructive Nonsmooth Analysis and Related Topics (Dedicated to the Memory of V. F. Demyanov) (CNSA). IEEE, 2017. P. 303−306.
13. Стецюк П. И. Оптимальный по объему 2d-эллипсоид и его приложения. Тезисы докладов Международной конференции «Конструктивный негладкий анализ и смежные вопросы», посвященной памяти профессора В. Ф. Демьянова. Часть II. СПб.: Издательство ВВМ, 2017. С. 173−176.
14. Стецюк П. І. Комп'ютерна програма «Octave-програма amsg2p — субградієнтний метод з кроком Поляка та перетворенням простору змінних». Свідоцтво про реєстрацію авторського права на твір № 88538. Україна. Міністерство освіти і науки. Державний департамент інтелектуальної власності. Дата реєстрації 11.05.2019.

3. УЗАГАЛЬНЕНИЙ МЕТОД ЕЛІПСОЇДІВ

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

Анотація. У рамках сімейства методів з розтягом простору описано загальну схему методу еліпсоїдів, частковим випадком якої є відомий метод еліпсоїдів Юдіна − Немировського − Шора. Наведено опис узагальненого методу еліпсоїдів і доведена теорема про його збіжність. Наведена H-форма узагальненого методу еліпсоїдів, де коригується додатно визначена симетрична матриця H. Описані правила побудови відсікаючих гіперплощин та критеріїв зупинки для задачі мінімізації опуклої функції, задачі опуклого програмування та задачі пошуку сідлової точки опукло-увігнутої функції. Описана алгоритмічна реалізація запропонованого Н. З. Шором методу еліпсоїдів у B-формі та відповідна програма **emshor** для знаходження точки мінімуму опуклої функції.

Annotation. Within the framework of a family of methods with space dilation, a general scheme of the ellipsoid method, which is a partial case of the well-known Yudin − Nemirovski − Shor ellipsoid method, is described. A description of the generalized ellipsoid method is given and theorem on its convergence is proved. The H-form of the generalized ellipsoid method is given, where a positively defined symmetric matrix H is updated. The rules for constructing cutting hyperplanes and stopping criteria for the problem of minimizing the convex function, the convex programming problem and the problem of finding the saddle point of a convex-concave function are described. The algorithmic realization of the proposed by N. Z. Shor ellipsoid method in B-form and the corresponding **emshor** program for finding the minimum point of the convex function are described.

3.1. Вступ

Наведено алгоритм з розтягом простору, який за певного вибору коефіцієнта розтягу є методом описаних еліпсоїдів [1]. Показано, що його частковим випадком є метод еліпсоїдів Юдіна – Немировського – Шора. Описано застосування алгоритму для розв'язання задачі опуклого програмування та задачі пошуку сідлової точки опукло-ввігнутої функції.

Класичний метод еліпсоїдів вперше запропоновано в 1976 році Д. Б. Юдіним та А. С. Немировським [2]. Вони виходили зі схеми послідовних відсікань і

назвали метод еліпсоїдів модифікованим методом центрованих перерізів (ММЦП). Незалежно метод еліпсоїдів був перевідкритий у 1977 році Н. З. Шором у роботі [3], де метод еліпсоїдів представлений як частковий випадок субградієнтних методів з розтягом простору в напрямку субградієнта (початок розвитку субградієнтних методів припав на 1969–1970 рр.).

Н. З. Шор указав коефіцієнт розтягу простору й параметри регулювання кроку в напрямку нормованого антисубградієнта такими, що субградієнтний метод з розтягом простору сходився з геометричною швидкістю спадання об'єму еліпсоїда, в якому локалізована точка мінімуму опуклої функції, і таким чином отримав дуже прозоре обґрунтування (доведення) збіжності методу еліпсоїдів.

Розглянуто сімейство методів еліпсоїдів, яке представлено як метод з розтягом простору та певним способом регулювання кроку, що пов'язаний з переходом до центра наступного локалізуючого еліпсоїда, об'єм якого буде меншим, ніж об'єм попереднього еліпсоїда. Назвемо його узагальненим методом еліпсоїдів. Розглянуто його застосування, доведена його збіжність з геометричною швидкістю спадання об'єму локалізуючого еліпсоїда, а також вказані коефіцієнти для двох часткових випадків узагальненого методу еліпсоїдів.

У підрозділі 3.2 наведено опис узагальненого методу еліпсоїдів і доведена теорема про його збіжність, сформульована в [4, 5].

У підрозділі 3.3 наведена H-форма узагальненого методу еліпсоїдів, де, як і в ММЦП, коригується додатно визначена симетрична матриця H. Тут показано, що відомий метод еліпсоїдів Юдіна – Немировського – Шора є частковим випадком узагальненого методу еліпсоїдів.

У підрозділі 3.4 описані правила побудови відсікаючих векторів та критеріїв зупинки для задачі мінімізації опуклої функції, задачі опуклого програмування та задачі пошуку сідлової точки опукло-увігнутої функції.

У підрозділі 3.5 описана octave-програма **emshor** [6], яка реалізує запропонований Н. З. Шором метод еліпсоїдів у B-формі (коригується несиметрична матриця B) для знаходження точки мінімуму опуклої функції $f(x)$.

3.2. Загальна схема методу еліпсоїдів

Задача. На $E^n (n \geq 2)$ задано векторне поле $g(x), g(x) \in E^n$. Необхідно знайти точку x^*, таку, що $(g(x), x - x^*) \geq 0$ для всіх $x \in E^n$. Вважається, що x^* існує та $g(x) \neq 0$ для $x \neq x^*$. Тут E^n — евклідовий простір розмірності n зі скалярним добутком (x, y).

Цю задачу можна розв'язати узагальненим методом еліпсоїдів, який є алгоритмом з розтягом простору, де коефіцієнт розтягу α задовольняє нерівності $\alpha + \frac{1}{\alpha} < 2\sqrt[n]{\alpha}$. Загальна схема методу еліпсоїдів має такий вигляд.

Ініціалізація. Вибираємо точку $x_0 \in E^n$ і радіус r_0 такими, щоб $\left\| B_0^{-1}(x_0 - x^*) \right\| \leq r_0$, де B_0 — $n \times n$-матриця. Перейдемо до наступної ітерації зі значеннями x_0, r_0, B_0.

Ітераційний процес. Нехай на k-й ітерації знайдені $x_k \in E^n$, r_k і $n \times n$-матриця B_k. Для переходу до $(k+1)$-ї ітерації виконаємо такі дії.

Крок 1. Обчислимо $g_k = g(x_k)$. Якщо $g_k = 0$, то ЗУПИНКА ($x^* = x_k$).

Крок 2. Обчислимо наступну точку

$$x_{k+1} := x_k - h_k B_k \xi_k, \text{ де } h_k = \frac{1}{2}\left(1 - \frac{1}{\alpha^2}\right) r_k, \xi_k = \frac{B_k^T g_k}{\left\| B_k^T g_k \right\|}.$$

Крок 3. Перерахуємо матрицю B_{k+1} і радіус r_{k+1}

$$B_{k+1} := B_k + \left(\frac{1}{\alpha} - 1\right)(B_k \xi_k)\xi_k^T, \quad r_{k+1} := \frac{1}{2}\left(\alpha + \frac{1}{\alpha}\right) r_k.$$

Крок 4. Переходимо до $(k+1)$-ї ітерації з x_{k+1}, r_{k+1} і B_{k+1}.

Теорема 3.1. Послідовність точок $\{x_k\}_{k=0}^{\infty}$, що генерується узагальненим методом еліпсоїдів, задовольняє нерівності

$$\left\| A_k (x_k - x^*) \right\| \le r_k, \quad k = 0,1,2,\ldots, \tag{3.1}$$

де $A_k = B_k^{-1}$. Відношення об'ємів еліпсоїдів $E_k = \left\{ x : \left\| A_k (x_k - x) \right\| \le r_k \right\}$ та $E_{k+1} = \left\{ x : \left\| A_{k+1} (x_{k+1} - x) \right\| \le r_{k+1} \right\}$, що локалізують точку x^*, є величиною сталою і рівною

$$q_n(\alpha) = \frac{vol(E_{k+1})}{vol(E_k)} = \frac{1}{\alpha} \left(\frac{1}{2} \left(\alpha + \frac{1}{\alpha} \right) \right)^n < 1, \quad k = 0,1,2,\ldots \tag{3.2}$$

Наведемо доведення теореми 3.1 подібно до того, як для методу еліпсоїдів це було зроблено Н. З. Шором [3]. Перш ніж перейти до доведення, дамо відношення для оператора розтягу простору

$$R_\alpha(\xi) = I_n + (\alpha - 1)\xi\xi^T, \quad \xi \in E^n, \quad \| \xi \| = 1, \tag{3.3}$$

які використані під час доведення теореми 3.1. Це будуть такі співвідношення:

$$R_\alpha^T(\xi) R_\alpha(\xi) = R_{\alpha^2}(\xi), \tag{3.4}$$

$$\det R_\alpha(\xi) = \alpha, \tag{3.5}$$

які випливають із властивостей оператора розтягу простору (3.3), див. [7], с. 68–69. Крім того, також використовуватимемо співвідношення

$$A_{k+1} = R_\alpha(\xi_k) A_k, \tag{3.6}$$

яке, враховуючи, що $\alpha = 1/\beta$, випливає з низки рівностей

$$A_{k+1} = B_{k+1}^{-1} = \left(B_k R_\beta(\xi_k) \right)^{-1} = R_\beta^{-1}(\xi_k) B_k^{-1} = R_{1/\beta}(\xi_k) A_k = R_\alpha(\xi_k) A_k.$$

Доведення. Доведення теореми 3.1 проводиться методом індукції по k. Для $k = 0$ нерівність (3.1) переходить у $\left\| A_0 (x_0 - x^*) \right\| \le r_0$, де $A_0 = B_0^{-1}$, і виконується за припущенням. Нехай нерівність (3.1) виконується для $k = \overline{k}$. Доведемо її справедливість для $k = \overline{k} + 1$.

Враховуючи відношення (3.4) і те, що з (3.6) випливає $A_{\overline{k}+1} = R_\alpha(\xi_{\overline{k}}) A_{\overline{k}}$, маємо такі рівності:

$$\| A_{\bar{k}+1}(x_{\bar{k}+1} - x^*) \|^2 = (A_{\bar{k}+1}(x_{\bar{k}+1} - x^*), A_{\bar{k}+1}(x_{\bar{k}+1} - x^*)) =$$
$$= (R_\alpha(\xi_{\bar{k}}) A_{\bar{k}}(x_{\bar{k}+1} - x^*), R_\alpha(\xi_{\bar{k}}) A_{\bar{k}}(x_{\bar{k}+1} - x^*)) = (A_{\bar{k}}(x_{\bar{k}+1} - x^*), R_\alpha^T(\xi_{\bar{k}}) R_\alpha(\xi_{\bar{k}}) A_{\bar{k}}(x_{\bar{k}+1} - x^*)) =$$
$$= (A_{\bar{k}}(x_{\bar{k}+1} - x^*), R_{\alpha^2}(\xi_{\bar{k}}) A_{\bar{k}}(x_{\bar{k}+1} - x^*)) = (A_{\bar{k}}(x_{\bar{k}+1} - x^*), (I + (\alpha^2 - 1)\xi_{\bar{k}}\xi_{\bar{k}}^T) A_{\bar{k}}(x_{\bar{k}+1} - x^*)) =$$
$$= (A_{\bar{k}}(x_{\bar{k}+1} - x^*), A_{\bar{k}}(x_{\bar{k}+1} - x^*)) + (\alpha^2 - 1)(\xi_{\bar{k}}, A_{\bar{k}}(x_{\bar{k}+1} - x^*))^2 =$$
$$= \| A_{\bar{k}}(x_{\bar{k}+1} - x^*) \|^2 + (\alpha^2 - 1)(\xi_{\bar{k}}, A_{\bar{k}}(x_{\bar{k}+1} - x^*))^2,$$

які запишемо у вигляді співвідношення:

$$\| A_{\bar{k}+1}(x_{\bar{k}+1} - x^*) \|^2 = \| A_{\bar{k}}(x_{\bar{k}+1} - x^*) \|^2 + (\alpha^2 - 1)(\xi_{\bar{k}}, A_{\bar{k}}(x_{\bar{k}+1} - x^*))^2. \qquad (3.7)$$

Далі розшифруємо обидва доданки в правій частині (3.7), для чого використаємо відношення

$$A_{\bar{k}}(x_{\bar{k}+1} - x^*) = A_{\bar{k}}(x_{\bar{k}} - x^*) - h_{\bar{k}}\xi_{\bar{k}}, \qquad (3.8)$$

яке, враховуючи, що $A_{\bar{k}} = B_{\bar{k}}^{-1}$ і наступна точка в узагальненому методі еліпсоїдів обчислюється за формулою з кроку 3, випливає з ланцюга рівностей

$$A_{\bar{k}}(x_{\bar{k}+1} - x^*) = A_{\bar{k}}(x_{\bar{k}} - h_{\bar{k}} B_{\bar{k}}\xi_{\bar{k}} - x^*) = A_{\bar{k}}(x_{\bar{k}} - x^*) - h_{\bar{k}} A_{\bar{k}} B_{\bar{k}}\xi_{\bar{k}} = A_{\bar{k}}(x_{\bar{k}} - x^*) - h_{\bar{k}}\xi_{\bar{k}}.$$

Перший доданок у правій частині (3.7) можна записати у вигляді рівності:

$$\| A_{\bar{k}}(x_{\bar{k}+1} - x^*) \|^2 = \| A_{\bar{k}}(x_{\bar{k}} - x^*) \|^2 - 2h_{\bar{k}}\left(A_{\bar{k}}(x_{\bar{k}} - x^*), \xi_{\bar{k}}\right) + h_{\bar{k}}^2, \qquad (3.9)$$

яка з урахуванням (3.8) і того, що $\| \xi_{\bar{k}} \| = 1$, випливає з низки рівностей

$$\| A_{\bar{k}}(x_{\bar{k}+1} - x^*) \|^2 = \| A_{\bar{k}}(x_{\bar{k}} - x^*) - h_{\bar{k}}\xi_{\bar{k}} \|^2 = \left(A_{\bar{k}}(x_{\bar{k}} - x^*) - h_{\bar{k}}\xi_{\bar{k}}, A_{\bar{k}}(x_{\bar{k}} - x^*) - h_{\bar{k}}\xi_{\bar{k}}\right) =$$
$$= \left(A_{\bar{k}}(x_{\bar{k}} - x^*), A_{\bar{k}}(x_{\bar{k}} - x^*)\right) - 2h_{\bar{k}}\left(A_{\bar{k}}(x_{\bar{k}} - x^*), \xi_{\bar{k}}\right) + h_{\bar{k}}^2(\xi_{\bar{k}}, \xi_{\bar{k}}) =$$
$$= \| A_{\bar{k}}(x_{\bar{k}} - x^*) \|^2 - 2h_{\bar{k}}\left(A_{\bar{k}}(x_{\bar{k}} - x^*), \xi_{\bar{k}}\right) + h_{\bar{k}}^2 \| \xi_{\bar{k}} \|^2 = \| A_{\bar{k}}(x_{\bar{k}} - x^*) \|^2 - 2h_{\bar{k}}\left(A_{\bar{k}}(x_{\bar{k}} - x^*), \xi_{\bar{k}}\right) + h_{\bar{k}}^2.$$

Враховуючи співвідношення (3.8), для квадрата скалярного добутку в другому доданку правої частини (3.7) маємо такий ланцюжок рівностей:

$$\left(A_{\bar{k}}(x_{\bar{k}+1} - x^*), \xi_{\bar{k}}\right)^2 = \left(A_{\bar{k}}(x_{\bar{k}} - x^*) - h_{\bar{k}}\xi_{\bar{k}}, \xi_{\bar{k}}\right)^2 =$$
$$= \left(\left(A_{\bar{k}}(x_{\bar{k}} - x^*), \xi_{\bar{k}}\right) - h_{\bar{k}}(\xi_{\bar{k}}, \xi_{\bar{k}})\right)^2 = \left(\left(A_{\bar{k}}(x_{\bar{k}} - x^*), \xi_{\bar{k}}\right) - h_{\bar{k}} \| \xi_{\bar{k}} \|^2\right)^2 =$$
$$= \left(\left(A_{\bar{k}}(x_{\bar{k}} - x^*), \xi_{\bar{k}}\right) - h_{\bar{k}}\right)^2 = \left(A_{\bar{k}}(x_{\bar{k}} - x^*), \xi_{\bar{k}}\right)^2 - 2h_{\bar{k}}\left(A_{\bar{k}}(x_{\bar{k}} - x^*), \xi_{\bar{k}}\right) + h_{\bar{k}}^2.$$

Отже, квадрат вказаного скалярного добутку можна записати у вигляді:

$$\left(A_{\bar{k}}(x_{\bar{k}+1}-x^*),\xi_{\bar{k}}\right)^2=\left(A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right)^2-2h_{\bar{k}}\left(A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right)+h_{\bar{k}}^2. \qquad (3.10)$$

Підставляючи (3.9) і (3.10) у (3.7), маємо

$$\| A_{\bar{k}+1}(x_{\bar{k}+1}-x^*)\|^2=\| A_{\bar{k}}(x_{\bar{k}+1}-x^*)\|^2+(\alpha^2-1)(\xi_{\bar{k}},A_{\bar{k}}(x_{\bar{k}+1}-x^*))^2=\| A_{\bar{k}}(x_{\bar{k}}-x^*)\|^2-$$
$$-2h_{\bar{k}}\left(A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right)+h_{\bar{k}}^2+(\alpha^2-1)\left(\left(A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right)^2-2h_{\bar{k}}\left(A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right)+h_{\bar{k}}^2\right)=$$
$$=\| A_{\bar{k}}(x_{\bar{k}}-x^*)\|^2-2\alpha^2h_{\bar{k}}\left(A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right)+(\alpha^2-1)\left(A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right)^2+\alpha^2h_{\bar{k}}^2=$$
$$=\| A_{\bar{k}}(x_{\bar{k}}-x^*)\|^2-\left(A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right)\left(2\alpha^2h_{\bar{k}}-(\alpha^2-1)\left(A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right)\right)+\alpha^2h_{\bar{k}}^2.$$

Звідси з урахуванням $h_k=\frac{1}{2}\left(1-\frac{1}{\alpha^2}\right)r_k$ маємо

$$\| A_{\bar{k}+1}(x_{\bar{k}+1}-x^*)\|^2=$$
$$=\| A_{\bar{k}}(x_{\bar{k}}-x^*)\|^2-(\alpha^2-1)\left(A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right)\left(r_{\bar{k}}-\left(A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right)\right)+\frac{(\alpha^2-1)^2}{4\alpha^2}r_{\bar{k}}^2. \qquad (3.11)$$

Далі для оцінки знаку добутку

$$\left(A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right)\left(r_{\bar{k}}-\left(A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right)\right),$$

що входить до правої частини відношення (3.11), оцінимо знаки обох його співмножників. Перший співмножник буде невід'ємним. Враховуючи, що $(x-x^*,g(x))\geq 0$ для всіх $x\in E^n$, його легко оцінити таким чином:

$$\left(A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right)=\left(A_{\bar{k}}(x_{\bar{k}}-x^*),\frac{B_{\bar{k}}^Tg(x_{\bar{k}})}{\| B_{\bar{k}}^Tg(x_{\bar{k}})\|}\right)=\frac{1}{\| B_{\bar{k}}^Tg(x_{\bar{k}})\|}\left(A_{\bar{k}}(x_{\bar{k}}-x^*),B_{\bar{k}}^Tg(x_{\bar{k}})\right)=$$
$$=\frac{1}{\| B_{\bar{k}}^Tg(x_{\bar{k}})\|}\left(B_{\bar{k}}A_{\bar{k}}(x_{\bar{k}}-x^*),g(x_{\bar{k}})\right)=\frac{1}{\| B_{\bar{k}}^Tg(x_{\bar{k}})\|}\left(x_{\bar{k}}-x^*,g(x_{\bar{k}})\right)\geq 0.$$

Враховуючи, що

$$0\leq\left(A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right)\leq\left\|A_{\bar{k}}(x_{\bar{k}}-x^*)\right\|\leq r_{\bar{k}},$$

другий співмножник оцінюється таким чином:

$$r_{\bar{k}}-\left(A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right)\geq 0.$$

З невід'ємності обох співмножників випливає, що

$$(\alpha^2-1)\left(A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right)\left(r_{\bar{k}}-\left(A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right)\right)\geq 0. \tag{3.12}$$

Далі, враховуючи (3.12) і те, що $\| A_{\bar{k}}(x_{\bar{k}}-x^*)\|\leq r_{\bar{k}}$, відношення (3.11) перепишемо у вигляді

$$\| A_{\bar{k}+1}(x_{\bar{k}+1}-x^*)\|^2\leq\| A_{\bar{k}}(x_{\bar{k}}-x^*)\|^2+\frac{(\alpha^2-1)^2}{4\alpha^2}r_{\bar{k}}^2\leq r_{\bar{k}}^2+\frac{(\alpha^2-1)^2}{4\alpha^2}r_{\bar{k}}^2=$$
$$=\left(\frac{\alpha^4+2\alpha^2+1}{4\alpha^2}\right)r_{\bar{k}}^2,$$

звідки маємо нерівність

$$\| A_{\bar{k}+1}(x_{\bar{k}+1}-x^*)\|^2\leq r_{\bar{k}}^2\left(\frac{1}{2}\left(\alpha+\frac{1}{\alpha}\right)\right)^2=r_{\bar{k}+1}^2,$$

з якої випливає справедливість нерівності (3.1) при $k=\bar{k}+1$.

Множина точок x, що задовольняє нерівності $\| A_k(x_k-x)\|\leq r_k$, є еліпсоїдом E_k, що містить точку x^*. Еліпсоїд E_k має об'єм

$$vol(E_k)=\frac{v_0 r_k^n}{\det A_k}, \tag{3.13}$$

де v_0 — об'єм одиничної n-вимірної кулі, $\det A_k$ — визначник матриці A_k.

Отже, швидкість збіжності узагальненого методу еліпсоїдів визначатиметься відношенням об'єму еліпсоїда E_{k+1}, що локалізує x^* на $(k+1)$-й ітерації, до об'єму еліпсоїда E_k, що локалізує x^* на k-й ітерації.

Для A_{k+1}, згідно з (3.6), маємо $A_{k+1}=R_\alpha(\xi_k)A_k$. Отже, для визначників цих матриць, враховуючи їхню невиродженість, справедливе співвідношення $\det A_{k+1}=\det R_\alpha(\xi_k)\det A_{k+1}$. Враховуючи формули (3.13) і (3.6), знайдемо коефіцієнт зменшення об'єму

$$q_n(\alpha)=\frac{vol(E_{k+1})}{vol(E_k)}=\frac{v_0 r_{k+1}^n\det A_k}{v_0 r_k^n\det A_{k+1}}=\left(\frac{r_{k+1}}{r_k}\right)^n\frac{\det A_k}{\det R_\alpha(\xi_k)\det A_k}=\left(\frac{r_{k+1}}{r_k}\right)^n\frac{1}{\alpha}=\frac{1}{\alpha}\left(\frac{1}{2}\left(\alpha+\frac{1}{\alpha}\right)\right)^n,$$

звідси, якщо коефіцієнт розтягу α задовольняє нерівності $\alpha + \frac{1}{\alpha} < 2\sqrt[n]{\alpha}$, маємо $q_n(\alpha) = \frac{1}{\alpha}\left(\frac{1}{2}\left(\alpha + \frac{1}{\alpha}\right)\right)^n < 1$. Теорема 3.1 доведена.

3.3. H-форма узагальненого методу еліпсоїдів

Узагальнений метод еліпсоїдів можна записати в H-формі (як ММЦП Юдіна – Неміровського) за допомогою додатно визначеної симетричної матриці $H_k = B_k B_k^T$. Для коефіцієнта розтягу α, який задовольняє нерівності $\alpha + \frac{1}{\alpha} < 2\sqrt[n]{\alpha}$, H-форма загального методу еліпсоїдів має такий вигляд.

Ініціалізація. Вибираємо точку $x_0 \in E^n$ і радіус r_0 такими, щоб $(x_0 - x^*)^T H_0^{-1}(x_0 - x^*) \le r_0^2$, де H_0 — додатно визначена симетрична $n \times n$-матриця. Перейдемо до наступної ітерації зі значеннями x_0, r_0, H_0.

Ітераційний процес. Нехай на k-й ітерації знайдені $x_k \in E^n$, r_k і $n \times n$-матриця H_k. Для переходу до $(k+1)$-ї ітерації виконуємо такі дії.

Крок 1. Обчислюємо $g_k = g(x_k)$. Якщо $g_k = 0$, то ЗУПИНКА($x^* = x_k$).

Крок 2. Обчислюємо наступну точку

$$x_{k+1} := x_k - h_k \frac{H_k g_k}{\sqrt{g_k^T H_k g_k}}, \text{ де } h_k = \frac{1}{2}\left(1 - \frac{1}{\alpha^2}\right) r_k \ .$$

Крок 3. Перерахуємо матрицю H_{k+1} і радіус r_{k+1}:

$$H_{k+1} := H_k + \left(\frac{1}{\alpha^2} - 1\right)\frac{H_k g_k g_k^T H_k}{g_k^T H_k g_k}, \quad r_{k+1} := \frac{1}{2}\left(\alpha + \frac{1}{\alpha}\right) r_k .$$

Крок 4. Переходимо до $(k+1)$-ї ітерації з x_{k+1}, r_{k+1} і H_{k+1}.

Для H-форми узагальненого методу еліпсоїдів формула для перерахунку наступного наближення x_{k+1} (крок 2) випливає зі справедливості такої низки співвідношень:

$$B_k \frac{B_k^T g_k}{\left\| B_k^T g_k \right\|} = \frac{B_k B_k^T g_k}{\sqrt{\left(B_k^T g_k\right)^T B_k g_k}} = \frac{H_k g_k}{\sqrt{\left(g_k\right)^T B_k B_k^T g_k}} = \frac{H_k g_k}{\sqrt{\left(g_k\right)^T H_k g_k}}.$$

Формула для перерахунку додатно визначеної симетричної матриці H_{k+1} (крок 3) випливає з такої низки співвідношень:

$$H_{k+1} = B_{k+1} B_{k+1}^T = B_k R_\beta(\xi_k)\left(B_k R_\beta(\xi_k)\right)^T = B_k R_\beta(\xi_k) R_\beta^T(\xi_k) B_k^T = B_k R_\beta(\xi_k) R_\beta(\xi_k) B_k^T =$$
$$= B_k R_{\beta^2}(\xi_k) B_k^T = B_k (I_n + (\beta^2 - 1)\xi_k \xi_k^T) B_k^T = B_k B_k^T + (\beta^2 - 1) B_k \xi_k \xi_k^T B_k^T =$$
$$= H_k + (\beta^2 - 1)\frac{B_k B_k^T g_k g_k^T B_k B_k^T}{\left\| B_k^T g_k \right\|^2} = H_k + (\beta^2 - 1)\frac{H_k g_k g_k^T H_k}{\left(B_k^T g_k\right)^T B_k^T g_k} = H_k + (\beta^2 - 1)\frac{H_k g_k g_k^T H_k}{g_k^T B_k B_k^T g_k} =$$
$$= H_k + (\beta^2 - 1)\frac{H_k g_k g_k^T H_k}{g_k^T H_k g_k},$$

де $\beta = 1/\alpha$.

Теорема 3.2. Послідовність точок $\{x_k\}_{k=0}^{\infty}$, що генеруються H-формою узагальненого методу еліпсоїдів, задовольняє нерівності

$$(x_k - x^*)^T H_k^{-1}(x_k - x^*) \le r_k^2, \quad k = 0,1,2,\ldots, \tag{3.14}$$

а відношення об'ємів еліпсоїдів $E_k = \left\{x : (x_k - x)^T H_k^{-1}(x_k - x) \le r_k^2\right\}$ і $E_{k+1} = \left\{x : (x_{k+1} - x)^T H_{k+1}^{-1}(x_{k+1} - x) \le r_{k+1}^2\right\}$, що локалізують точку x^*, є величиною сталою і рівною

$$q_n(\alpha) = \frac{vol(E_{k+1})}{vol(E_k)} = \frac{1}{\alpha}\left(\frac{1}{2}\left(\alpha + \frac{1}{\alpha}\right)\right)^n < 1, \quad k = 0,1,2,\ldots \tag{3.15}$$

У теоремах 3.1 і 3.2 співвідношення (3.2) та (3.15) означають, що метод еліпсоїдів сходиться (по об'єму локалізації точки x^*) зі швидкістю геометричної прогресії зі знаменником $q_n(\alpha) < 1$. Величина знаменника залежить від вибраного значення α, що задовольняє нерівності $\alpha + \frac{1}{\alpha} < 2\sqrt[n]{\alpha}$.

Найменший знаменник прогресії реалізується у методі еліпсоїдів Юдіна – Немировського − Шора. Йому відповідає коефіцієнт розтягу $\alpha_1 = \sqrt{\frac{n+1}{n-1}}$, і досягається він у точці мінімуму функції $q_n(\alpha)$ по α. Близький до найменшого знаменник прогресії реалізується у наближеному методі еліпсоїдів [8], і йому відповідає коефіцієнт розтягу $\alpha_2 = \sqrt{1+\frac{1}{n^2}} + \frac{1}{n}$. Він досягається у точці мінімуму функції $Q_n(\alpha)$, яка апроксимує зверху функцію $q_n(\alpha)$ згідно з таким співвідношенням:

$$q_n(\alpha) = \frac{1}{\alpha}\left(\frac{1}{2}\left(\alpha + \frac{1}{\alpha}\right)\right)^n = \frac{1}{\alpha}\left(1 + \frac{1}{2}\left(\alpha + \frac{1}{\alpha} - 2\right)\right)^n \le \frac{1}{\alpha}\exp\left\{\frac{n}{2}\left(\alpha + \frac{1}{\alpha} - 2\right)\right\} = Q_n(\alpha).$$

При великих значеннях n знаменники геометричної прогресії в обох методах апроксимуються зверху близькими величинами $q^*(n) = 1 - \frac{1}{2n}$ і $Q^*(n) = 1 - \frac{1}{2n} + \frac{1}{2n^2}$.

3.4. Задачі для узагальненого методу еліпсоїдів

Нижче наведений опис деяких задач із [9], для розв'язку яких можна використовувати узагальнений метод еліпсоїдів.

1. *Задача безумовної мінімізації опуклої функції.* Нехай $f(x)$ — опукла функція, де $x \in E^n$. Її мінімальне значення позначатимемо як $f^* = f(x^*)$ і, не обмежуючи загальності, вважатимемо, що точка x^* — єдина точка мінімуму. Нехай є апріорна інформація, що точка x^* знаходиться у кулі $S(x_0, R)$. Тоді, якщо векторне поле визначене за формулою $g(x) = g_f(x)$, де $g_f(x)$ — субградієнт функції $f(x)$ у точці x, то для нього виконуватиметься нерівність

$$\left(x - x^*, g(x)\right) = \left(x - x^*, g_f(x)\right) \ge f(x) - f(x^*) = f(x) - f^* \ge 0, \quad \forall x \in E^n. \qquad (3.16)$$

Отже, для знаходження точки x^* можна використовувати метод еліпсоїдів, вказавши стартову точку x_0, початковий радіус $r_0 = R$ і матрицю $B_0 = I_n$, де I_n — одинична $n \times n$-матриця. Як критерій зупинки можна використовувати умову $r_k \left\| B_k^T g_f(x_k) \right\| \le \varepsilon$, яка при довільному малому ε дозволяє знайти точку $x_\varepsilon^* = x_k$, для якої $f(x_\varepsilon^*) - f^* \le \varepsilon$. Це випливає з нерівності

$$r_k \ge \left\| B_k^{-1}(x_k - x^*) \right\| \ge \left(B_k^{-1}(x_k - x^*), \frac{B_k^T g_f(x_k)}{\left\| B_k^T g_f(x_k) \right\|} \right) = \frac{(x_k - x^*, g_f(x_k))}{\left\| B_k^T g_f(x_k) \right\|} \ge \frac{f(x_k) - f^*}{\left\| B_k^T g_f(x_k) \right\|},$$

яка справедлива для опуклої функції $f(x)$ з урахуванням умови (3.16).

2. *Загальна задача опуклого програмування.* Знайти

$$f_0^* = f_0(x^*) = \min_{x \in E^n} f_0(x) \tag{3.17}$$

при обмеженнях

$$f_i(x) \le 0, \quad i = 1, 2, \ldots, m, , \tag{3.18}$$

де $f_i(x)$ — опуклі функції, визначені на E^n, $g_i(x)$ — відповідні субградієнти, $i = 0, 1, \ldots, m$. Нехай відомо, що оптимальна точка x^* існує та знаходиться у кулі $S(x_0, R)$ (формально до системи обмежень (3.18) можна додати обмеження $\left\| x - x_0 \right\| \le R$), і буде виконана умова Слейтера для задачі (3.17)–(3.18).

Розглянемо векторне поле $g(x)$, побудоване таким чином:

$$g(x) = \begin{cases} g_0(x), & \text{якщо } \max\limits_{1 \le i \le m} f_i(x) \le 0, \\ g_{i*}(x), & \text{якщо } \max\limits_{1 \le i \le m} f_i(x) = f_{i*}(x) > 0. \end{cases} \tag{3.19}$$

Покажемо, що $(g(x), x - x^*) \ge 0$ при всіх $x \in E^n$. Якщо $\max\limits_{1 \le i \le m} f_i(x) \le 0$, то $g(x) = g_0(x)$,

$$(g(x), x - x^*) = (g_0(x), x - x^*) \ge f_0(x) - f_0(x^*) \ge 0.$$

Якщо $\max\limits_{1\le i\le m} f_i(x) > 0$, то $g(x) = g_{i*}(x)$, причому $f_{i*}(x) > 0$, $f_{i*}(x^*) \le 0$,

$$(g(x), x - x^*) = (g_{i*}(x), x - x^*) \ge f_{i*}(x) - f_{i*}(x^*) \ge 0.$$

Таким чином, справедлива нерівність $(g(x), x - x^*) \ge 0$ при всіх $x \in E^n$.

Отже, обчислюючи $g(x)$ за формулою (3.19), для локалізації x^* у задачі (3.17)–(3.18) можна використовувати узагальнений метод еліпсоїдів. Зауважимо, що цей результат не зміниться, якщо в другій формулі (3.19) замість $g_{i*}(x)$ брати $g_{\bar{i}}$, де $\bar{i}$ — довільний індекс, для якого $f_{\bar{i}}(x) > 0$. ЗУПИНКА за умови $r_k \left\| B_k^T g_0(x_k) \right\| \le \varepsilon$ дозволяє знайти точку $x_\varepsilon^* = x_k$, для якої $f_0(x_\varepsilon^*) - f_0^* \le \varepsilon$, що випливає зі справедливості нерівності

$$r_k \ge \left\| B_k^{-1}(x_k - x^*) \right\| \ge \left(B_k^{-1}(x_k - x^*), \frac{B_k^T g_0(x_k)}{\left\| B_k^T g_0(x_k) \right\|} \right) = \frac{(x_k - x^*, g_0(x_k))}{\left\| B_k^T g_0(x_k) \right\|} \ge \frac{f_0(x_k) - f_0^*}{\left\| B_k^T g_0(x_k) \right\|}$$

для опуклої функції $f_0(x)$.

3. *Задача про сідлову точку.* Нехай задана опукло-ввігнута функція $f(x, y)$ двох векторних змінних $x \in E^n$, $y \in E^m$, $z = \{x, y\} \in E^n \times E^m \equiv E^{n+m}$, z^* — сідлова точка цієї функції, z_0 – задане початкове наближення, та відомо, що $\left\| z_0 - z^* \right\| \le R$.

Розглянемо псевдоградієнтну множину $G(z) = G_f^x(x, y) \times (-G_f^y(x, y))$, де $G_f^x(x, y)$ — множина часткових субградієнтів функції $f(x, y)$, що розглядається як функція від x при фіксованому y; $G_f^y(x, y)$ — множина часткових суперградієнтів функції $f(x, y)$ по y при фіксованому x. Нехай векторне поле $g(z)$ побудоване таким чином:

$$g(z) = \{g_f^x(z), -g_f^y(z)\}, \quad g_f^x(z) \in G_f^x(z), \quad g_g^y(z) \in G_f^y(z). \tag{3.20}$$

Із визначення сідлової точки випливає, що $f(x,y^*) \geq f(x^*,y^*) \geq f(x^*,y)$. Тому

$$0 \leq f(x,y^*) - f(x^*,y) = f(x,y^*) - f(x,y) + f(x,y) - f(x^*,y) \leq$$
$$\leq (g_f^x(z), x - x^*) - (g_g^y(z), y - y^*) = (g(z), z - z^*),$$

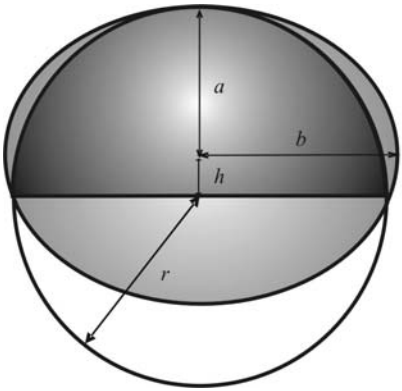
звідки випливає, що $(g(z), z - z^*) \geq 0$ при всіх $z \in E^{n+m}$. Таким чином, обчислюючи $g(z)$ за формулою (3.20), для локалізації сідлової точки z^* можна застосовувати узагальнений метод еліпсоїдів.

Узагальнений метод еліпсоїдів також має низку інших застосувань, наприклад, для розв'язання координуючих негладких задач невеликих розмірностей, які присутні в схемах декомпозиції (за обмеженнями, за змінними), у спеціальних опуклих задачах з невеликою кількістю змінних при параметрично заданому сімействі обмежень та ін.

Головне — потрібно визначити правило побудови відсікаючих векторів, які локалізують шукану точку, та умову зупинки ітераційного процесу в узагальненому методі еліпсоїдів.

3.5. Алгоритм Шора та програма emshor

Метод еліпсоїдів Юдіна – Немировського – Шора базується на використанні в E^n еліпсоїда мінімального об'єму, який містить напівкулю, отриману в результаті перетину n-вимірної кулі та півпростору, який проходить через її центр. Цей еліпсоїд має сплюснуту форму в напрямку нормалі до гіперплощини, яка проходить через центр кулі радіуса r. Його параметри (наведені на рис. 3.1) такі: a — довжина меншої півосі в напрямку нормалі, яка визначає напівкулю; b — довжина більшої півосі (кількість таких півосей дорівнює $n-1$); h — відстань від центра кулі до центра еліпсоїда в напрямку меншої з його півосей.



$$a = r\frac{n}{n+1}$$

$$b = r\frac{n}{\sqrt{n^2 - 1}}$$

$$h = r\frac{1}{n+1}$$

Рис. 3.1. Еліпсоїд мінімального об'єму, який містить півкулю в E^n

Ітерація методу еліпсоїдів полягає в переході від поточного еліпсоїда до наступного з постійним коефіцієнтом зменшення їхніх об'ємів. Цей коефіцієнт визначається відношенням об'єму еліпсоїда з півосями a та b до об'єму кулі радіуса r в E^n і може бути записаний у вигляді

$$q_n = \left(\frac{a}{r}\right)\left(\frac{b}{r}\right)^{n-1} = \frac{n}{n+1}\left(\frac{n}{\sqrt{n^2-1}}\right)^{n-1} < 1. \qquad (3.21)$$

Для нього показано, що

$$q_n < \exp\left\{-\frac{1}{2n}\right\} < 1; \qquad (3.22)$$

отже, при великих n коефіцієнт зменшення об'єму апроксимується асимптотичною формулою

$$q_n \approx 1 - \frac{1}{2n}. \qquad (3.23)$$

Тому, для зменшення об'єму еліпсоїда, який локалізує розв'язок задачі, в 10 разів, потрібно зробити K ітерацій, де

$$K = -\frac{\ln 10}{\ln q_n} \approx (2\ln 10)n \approx 4.6n. \qquad (3.24)$$

Щоб застосувати метод еліпсоїдів для знаходження розв'язку x^* задачі мінімізації опуклої функції або задачі опуклого програмування (3.17)–(3.18), потрібно вибрати початковий радіус кулі, яка локалізує точку x^*.

Опишемо його алгоритмічну реалізацію у B-формі, запропонованій Н. З. Шором, для знаходження точки x_ε^* — наближення до точки x^* мінімуму опуклої функції $f(x)$.

Алгоритм Шора для знаходження x_ε^*. Вхідним параметром алгоритму є величина ε_f — точність, з якою потрібно знайти значення $f^* = f(x^*)$. Алгоритм Шора для знаходження точки x_ε^* має такий вигляд.

Ініціалізація. Покладемо стартову точку $x_0 \in E^n$ і початковий радіус r_0 такими, щоб $\|x_0 - x^*\| \le r_0$. Введемо до розгляду $n \times n$-матрицю B і покладемо $B_0 := I_n$, де I_n — одинична $n \times n$-матриця. Перейдемо до першої ітерації зі значеннями x_0, r_0 і B_0.

Нехай на k-й ітерації знайдено значення $x_k \in E^n$, r_k, B_k. Перехід до $(k+1)$-ї ітерації полягає у виконанні такої послідовності дій.

Крок 1. Обчислимо $f(x_k)$ та $g(x_k)$ — субградієнт функції $f(x_k)$. Якщо $\|B_k^T g(x_k)\| r_k \le \varepsilon_f$, то ЗУПИНКА: $k^* = k$ і $x_\varepsilon^* = x_k$". Інакше переходимо до кроку 2.

Крок 2. Покладемо $\xi_k := \dfrac{B_k^T g(x_k)}{\|B_k^T g(x_k)\|}$.

Крок 3. Обчислимо чергову точку

$$x_{k+1} := x_k - h_k B_k \xi_k, \quad \text{де} \quad h_k = \frac{1}{n+1} r_k.$$

Крок 4. Обчислимо

$$B_{k+1} := B_k + \left(\sqrt{\frac{n-1}{n+1}} - 1 \right) (B_k \xi_k) \xi_k^T \quad \text{та} \quad r_{k+1} := r_k \frac{n}{\sqrt{n^2-1}}.$$

Крок 5. Перейдемо до $(k+1)$-ї ітерації зі значеннями x_{k+1}, r_{k+1}, B_{k+1}. Збіжність алгоритму забезпечує наступна теорема.

Теорема 3.3. Послідовність точок $\{x_k\}_{k=0}^{k^*}$, що генерується алгоритмом Шора, задовольняє нерівності

$$\left\|B_k^{-1}(x_k - x^*)\right\| \le r_k, \quad k = 0,1,2,\ldots,k^*.$$

На кожній ітерації k, де $1 \le k \le k^*$, відношення об'ємів еліпсоїдів $E_k = \left\{x : \left\|B_k^{-1}(x_k - x)\right\| \le r_k\right\}$ та $E_{k-1} = \left\{x : \left\|B_k^{-1}(x_{k-1} - x)\right\| \le r_{k-1}\right\}$, що локалізують x^*, є величиною сталою та рівною

$$q_n = \frac{vol(E_k)}{vol(E_{k-1})} = \frac{n}{n+1}\left(\frac{n}{\sqrt{n^2-1}}\right)^{n-1} < \exp\left\{-\frac{1}{2n}\right\} < 1.$$

Алгоритм Шора для знаходження x_ε^* реалізовано у вигляді Octave-програми **emshor**, код якої з англомовними коментарями наведено нижче.

```
# Octave-function emshor (P.Stetsyuk, September 11, 2017)
# Input parameters:
#    calcfg - name of the function calcfg(x)
#             for calculation of f and g
#    x0 - the starting point, x0(1:n)
#    rad - radius of the ball localizing the minimum point
#    epsf, maxitn - stop parameters
#    intp - print information every intp iteration
# Output parameters:
#    x - a minimum point, which was found by the program, x(1:n)
#    f - the value of the function f at the point x
#    itn - the number of iterations used by the program
#    nfg - the number of function calcfg calls
#    istop - exit code (1 = eps, 4 = maxitn)
function [x,f,itn,nfg,ist]=emshor(calcfg,x0,rad,       #row01
                                  epsf,maxitn,intp);
dn=double(length(x0)); beta=sqrt((dn-1.d0)/(dn+1.d0));  #row02
```

```
x=x0; radn=rad; B=eye(length(x)); nfg=0;                #row03
for (itn = 0:maxitn)                                    #row04
   [f, g1] = calcfg(x); if(f<inf) nfg=nfg+1; endif      #row05
   g=B'*g1; dg=norm(g);                                 #row06
   if(radn*dg < epsf) ist = 1;  return; endif           #row07
   xi=(1.d0/dg)*g; dx = B * xi; hs=radn/(dn+1.d0);      #row08
   x -= hs * dx;  B += (beta - 1) * B * xi * xi';       #row09
   radn=radn/sqrt(1.d0-1.d0/dn)/sqrt(1.d0+1.d0/dn);     #row10
   if(mod(itn,intp)==0)                                 #row11
     printf("itn %4d  f %14.6e  nfg %4d\n",itn,f,nfg); #row12
   endif                                                #row13
endfor                                                  #row14
ist = 4;                                                #row15
endfunction
```

Відзначимо, що, якщо $\left\|x_0 - x^*\right\| \le r_0$, програма **emshor** обов'язково закінчує свою роботу виконанням однієї з умов: (1) знайдена точка x_ε^* — така, що $f(x_\varepsilon^*) - f^* \le \varepsilon_f$ (**ist=1**), (2) **maxitn** ітерацій виявилося недостатньо (**ist = 4**) [10].

Роботу програми продемонструємо на тестовому прикладі, який полягає у мінімізації кусочно-лінійної функції вигляду

$$f(x) = \sum_{i=1}^{n} t_i \left|x_i - 1\right|, \quad f^* = f(x^*) = 0, \quad x^* = (1,1,\ldots,1)^T, \tag{3.25}$$

де $|a|$ — абсолютна величина числа a, t_i — задані коефіцієнти при $|x_i - 1|$, $i = 1,\ldots,100$. Яружність функції (3.25) залежить від відношення максимального коефіцієнта t_i до мінімального. В тестових експериментах використовувалися три види коефіцієнтів: коефіцієнту 2^{i-1} відповідають найбільш яружні функції, коефіцієнту $\left(5/6\right)^{i-1}$ — менш яружні, третій коефіцієнт — $t_i = i$ — ілюструє випадок, коли функція (3.25) не є яружною.

Результати роботи програми **emshor** наведено в таблиці 3.1 для трьох випадків коефіцієнтів функції (3.25).

Табл. 3.1

$f(x)=\sum_{i=1}^{n}\left(5/6\right)^{i-1}\lvert x_i-1\rvert$									
	$\varepsilon_f=10^{-3}$			$\varepsilon_f=10^{-6}$			$\varepsilon_f=10^{-9}$		
n	itn	f	time	itn	f	time	itn	f	time
5	452	1.5e – 4	6e – 3	785	1.7e – 7	0.1	1118	3.3e – 11	0.2
10	2015	3.7e – 6	0.3	3382	5.5e – 8	0.5	4693	8.4e – 11	0.7
15	4917	6.2e – 5	0.7	8104	2.4e – 9	1.2	11170	7.1e – 12	1.8
20	9507	4.8e – 5	1.5	14826	6.6e – 9	2.3	20544	1.3e – 12	3.3
$f(x)=\sum_{i=1}^{n}2^{i-1}\lvert x_i-1\rvert$									
	$\varepsilon_f=10^{-3}$			$\varepsilon_f=10^{-6}$			$\varepsilon_f=10^{-9}$		
n	itn	f	time	itn	f	time	itn	f	time
5	473	2.5e – 5	0.07	842	6.1e – 8	0.1	1164	2.0e – 11	0.2
10	512	3.0e – 5	0.3	3819	3.3e – 8	0.5	5225	7.5e – 11	0.8
15	6574	6.1e – 5	1.0	9712	6.2e – 8	1.5	12809	3.8e – 11	2.0
20	13322	4.8e – 5	2.1	18916	4.5e – 8	3.0	23397	3.3e – 11	3.7
$f_1=\sum_{i=1}^{n}i\lvert x_i-1\rvert$									
	$\varepsilon_f=10^{-3}$			$\varepsilon_f=10^{-6}$			$\varepsilon_f=10^{-9}$		
n	itn	f	time	int	f	time	itn	f	time
5	310	4.0e – 5	0.04	492	1.0e – 8	0.07	768	2.2e – 11	0.1
10	1749	5.3e – 5	0.2	2895	9.3e – 8	0.5	4007	8.8e – 11	0.6
15	4535	6.0e – 5	0.7	7293	6.1e – 8	1.2	9981	5.7e – 11	1.5
20	8698	4.7e – 5	1.4	13794	4.5e – 8	2.1	18838	4.1e – 11	3.0

Були проведені розрахунки для $n=5,\ 10,\ 15,\ 20$ при трьох значеннях $\varepsilon_f=10^{-3},\ 10^{-6},\ 10^{-9}$. Якщо $n=20$, то коефіцієнти 2^{i-1} утворюють геометричну прогресію з показником $q=2$, де мінімальний коефіцієнт дорівнює $(2)^0=1$, а максимальний — $(2)^{19}\approx 5.24288\text{e+}05$. Обчислення проводились на комп'ютері Pentium 3GHz у системі Windows7/32 за допомогою GNU Octave-версії 4.4.2.

За допомогою програми **emshor** можна знаходити досить точні наближення до точки мінімуму опуклої функції від декількох десятків змінних. Так, наприклад, якщо $n = 20$, то для цього потрібно декілька секунд (залежно від заданої точності) на сучасних персональних ЕОМ з використанням GNU Octave-версій 3.0.0 та вище.

3.6. Висновки

Для сімейства методів з розтягом простору описана загальна схема методу еліпсоїдів, яка дозволяє отримувати алгоритми описаних еліпсоїдів, що збігаються зі швидкістю геометричної прогресії, для знаходження деяких стаціонарних точок градієнтних полів. При цьому показник геометричної прогресії залежить лише від розмірності простору і не залежить від властивостей гра-дієнтного поля.

Використання цих алгоритмів доцільне при отриманні оцінок складності для алгоритмів розв'язання спеціальних класів задач математичного програмування, задач пошуку точок рівноваги, включаючи узагальнені рівноваги Неша [11, 12], а також для розв'язання систем нелінійних рівнянь з обмеженнями, варіаційних нерівностей і комплементарних задач. Якщо ж ці алгоритми пов'язані з розв'язанням складних координуючих підзадач з кількістю змінних, не більше десяти, то вони будуть ефективними і для практичних застосувань.

Крім того, побудовані еліпсоїди локалізації доцільно використовувати в загальній схемі побудови методів центрів тяжіння простих тіл [13]. Якщо їх використовувати в поєднанні з еліпсоїдами, що локалізують перетин кулі та двох півпросторів [14, 15, 16, 17], то швидкість збіжності алгоритмів описаних еліпсоїдів можна значно підвищити. Такі алгоритми матимуть теоретичну оцінку швидкості збіжності не гіршу, ніж у методі еліпсоїдів Юдіна – Немировського – Шора, а практичну — зможуть наблизити до ефективності *r*-алгоритмів Шора.

Список літератури

1. Стецюк П. И., Фесюк А. В., Хомяк О. Н. Обобщенный метод эллипсоидов. Кибернетика и системный анализ. 2018. № 4. С. 70–80.
2. Юдин Д. Б., Немировский А. С. Информационная сложность и эффективные методы решения выпуклых экстремальных задач. Экономика и математические методы. 1976. Вып. 2. С. 357–369.
3. Шор Н. З. Метод отсечения с растяжением пространства для решения задач выпуклого программирования. Кибернетика. 1977. № 1. С. 94–95.
4. Стецюк П. И. Общая схема метода эллипсоидов. Информационный бюллетень АМП № 13. Екатеринбург: УрО РАН, 2015. С. 59–60.
5. Стецюк П. И. Об одном обобщении классического метода эллипсоидов. Інформатика та системні науки (ІСН–2015): матеріали VI Всеукр. наук.-практ. конф. за міжнародною участю (м. Полтава, 19–21 березня 2015 р.). Полтава: ПУЕТ, 2015. С. 335–337.
6. Стецюк П. И. Метод эллипсоидов с берегов Днепра. Математичне та програмне забезпечення інтелектуальних систем: Тези доповідей XV Міжнародної наук.-практ. конференції MPZIS−2017, 22–24 листопада 2017 р. Дніпро: ДНУ, 2017. С. 185.
7. Шор Н. З. Методы минимизации недифференцируемых функций и их приложения. Киев: Наукова думка, 1979. 200 с. (English transl.: Shor N. Z. Minimization Methods for Non-Differetiable Functions. Berlin: Springer-Verlag, 1985. 178 p.)
8. Стецюк П. И. Приближенный метод эллипсоидов. Кибернетика и системный анализ. 2003. № 3. С. 141–146.
9. Шор Н. З. Новые направления в развитии методов негладкой оптимизации. Кибернетика. 1977. № 6. С. 87–91.

10. Стецюк П. І., Івлічев А. В. Метод еліпсоїдів та Octave-програма emshor. Міжнародний науковий симпозіум «Інтелектуальні рішення». Теорія прийняття рішень: праці Міжнар. школи-семінару, 15–20 квітня 2019 р. Ужгород: УНУ, 2019. С. 119–120.
11. Fischer A., Herrich M., Schönefeld K. Generalized Nash Equilibrium Problems – Recent Advances and Challenges. Pesquisa Operacional. 2014. Vol. 34. P. 521–558.
12. Daryina A. N., Izmailov A. F. Newton-type method for variational equilibrium problem. VIII Московская международная конференция по Исследованию Операций (ORM2016), Москва, 17–22 октября 2016. Труды. Т. 1. Москва: МАКС Пресс, 2016. С. 21–22.
13. Стецюк П. И. Метод центров тяжести простых тел. Кибернетика и системный анализ. 1996. № 5. С. 117–138.
14. Стецюк П. И. Методы эллипсоидов и r-алгоритмы. Кишинэу: Эврика, 2014. 488 с.
15. Stetsyuk P. I. 2d-Ellipsoid of optimal volume and its applications. In: L. N. Polyakova (ed.) Constructive Nonsmooth Analysis and Related Topics (Dedicated to the Memory of V. F. Demyanov) (CNSA). IEEE, 2017. P. 303–306.
16. Стецюк П. И. Оптимальный по объему 2d-эллипсоид и его приложения. Тезисы докладов Международной конференции «Конструктивный негладкий анализ и смежные вопросы», посвященной памяти профессора В. Ф. Демьянова. Часть II. СПб.: Издательство ВВМ, 2017. С. 173–176.
17. Стецюк П. И. r-алгоритмы и эллипсоиды. Кибернетика и системный анализ. 1996. № 1. С. 113–134.

Розділ 2. ЗАДАЧІ НА КОМБІНАТОРНИХ КОНФІГУРАЦІЯХ

4. ЗАДАЧІ ПРО МАТЕМАТИЧНІ СЕЙФИ

Г. П. Донець

Анотація. Робота присвячена дослідженню нового математичного об'єкта, який має назву математичний сейф. Задачі розглядаються для однотипних та багатотипних замків сейфів, які задаються на графах. Такі задачі зводяться до розв'язання системи лінійних рівнянь у класах лишків за певним модулем.

Annotation. The paper is devoted to the study of a new mathematical object, called the mathematical safe. Problems are considered for single-type and multi-type locks of safes, which are specified on the graphs. Such problems are reduced to the solution of the system of linear equations in the classes of residues by a certain module.

4.1. Вступ

Дослідження задач дискретної оптимізації є передумовою успішного моделювання важливих економічних, природних, соціальних та інших процесів. Наукові публікації протягом минулих двадцяти-тридцяти років у галузі дискретної оптимізації свідчать про необхідність і важливість подібних досліджень. Це пов'язано з тим, що за останній час зросла актуальність розв'язування таких задач під час ухвалення рішень у галузі управління та планування виробничих процесів, у задачах геометричного проектування, перспективного планування, теорії розкладів та інших, пов'язаних з вибором одного з можливих варіантів дії. Серед класу комбінаторних оптимізаційних задач особливе місце займають задачі на вершинно розташованих множинах. Аналіз використання результатів цих досліджень дозволяє дійти висновку про актуальність нових підходів і методів комбінаторної оптимізації.

Розглянемо нову перспективну тему, яка має пряме відношення до однієї з галузей теорії ігор. Маються на увазі задачі, де необхідно, з огляду на

початковий стан об'єкта, за певними правилами та мінімальними витратами досягти наперед заданого стану. Такі задачі зустрічаються у комп'ютерних іграх, і їх можна назвати в загальному як позиційні ігри, проте в математичній постановці вони звучать конкретно як задачі про математичний сейф.

4.2. Математичні сейфи на графах

Математичним сейфом називається система $Z=(z_1,z_2,...,z_N)$ взаємозалежних замків така, що, коли робиться поворот ключем в одному замку, то такий самий поворот робиться й у замках, пов'язаних із цим.

Математичний сейф найкраще задавати за допомогою орієнтованого графа, у якого замки є вершинами, а дуги вказують на їхній взаємозв'язок. Так дуга (z_i,z_j) вказує, що замок z_j пов'язаний із замком z_i, і з будь-яким поворотом ключа в замку z_i одночасно здійснюється поворот і в замку z_j. Замок z_i називається вхідним стосовно замка z_j. Будь-який замок може перебувати в одному з двох положень — відкритий чи закритий. Існують замки, для відкриття яких потрібно кілька поворотів ключа (0,1,2,…, *K*–1), кількість яких дорівнює *K*, що визначає загальне число станів замка. Замок відкритий, коли він у стані 0. В іншому стані замок закритий.

Необхідно розв'язати наступну задачу. З огляду на початковий стан сейфа $\boldsymbol{b} = (b_1, b_2,..., b_N)$, де $b_i \in (0,1,2,\ldots,K\text{-}1)$, знайти таку послідовність замків і число поворотів ключа в них, щоб сейф перейшов у положення відкритого, тобто стан усіх замків став рівним 0.

Наведемо математичну постановку такої задачі.

Нехай x_j, $j=1,2,...,N$, — необхідна кількість поворотів ключа під час розв'язування задачі в замку z_j. Для кожного замка z_j повинно виконуватися основне рівняння. Нехай $z_1,z_2,..,z_k$ — вхідні замки для замка z_j. Основне рівняння рівносильне твердженню: *сума чисел поворотів вхідних замків для замка* z_j, *плюс його число поворотів* x_j, *плюс його початковий стан* b_j *повинні бути рівна* $0 (\mathrm{mod}\ K)$. Це твердження запишеться

$$\sum_{i=1}^{k} x_i + x_j + b_j = 0(\mathrm{mod}K2), \quad j = 1,2,...,N. \tag{4.1}$$

Покажемо на прикладах, як розв'язувати такі задачі.

Приклад 1. Розглянемо найпростіший випадок, коли $K = 2$. Нехай вершини графа утворюють ланцюг, у якого всі ребра орієнтовані в одному напрямку, тобто ланцюг утворить шлях (рис. 4.1).

Занумеруємо його вершини від 1 до N, починаючи з вершини, що не має вхідної дуги. Номера вершин зазначені в кружечках, а під ними зазначені стани замків (вершин).

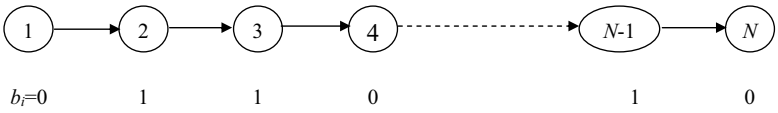


Рис. 4.1

Покладемо i=1. Якщо $b_i = 0$, то переходимо до вершини $i+1$. Якщо $b_i = 1$, то робимо поворот у замку z_i, переводячи його стан у 0, а b_{i+1} переходить у стан $(b_{i+1}+1)(\mathrm{mod}2)$. Після цього переходимо до вершини $i+1$. Зрештою приходимо до вершини N, стан якої переводимо в $b_N = 0$. У підсумку для всього графа одержимо стани $b_i = 0\ (1 \le i \le N)$.

Для розв'язування цієї задачі можна скористатися системою (4.1), що для цього прикладу після перенесення b_j у праву сторону має такий вигляд:

$$\left.\begin{array}{ccccccccccc}
x_1 & & & & & & & & . & \equiv -b_1 \\
x_1 + & x_2 & & & & & & & \cdot & \equiv -b_2 \\
 & x_2 + & x_3 & & & & & & \cdot & \equiv -b_3 \\
 & & x_3 + & x_4 & \cdot & \cdot & \cdot & & \cdot & \equiv -b_4 \\
 & & & \cdot & \cdot & \cdot & \cdot & & \cdot & \cdot \\
 & & & & & & x_{N-1} & + x_N & & \equiv -b_N
\end{array}\right\} \mathrm{mod}\,2. \tag{4.2}$$

Розв'язок цієї системи легко знаходиться шляхом послідовної підстановки значень x_i, i=1,2,...,$N-1$, у наступне рівняння.

Приклад 2. Розглянемо граф на рис. 4.2, що являє собою контур, на якому заданий сейф із замками для K=2.

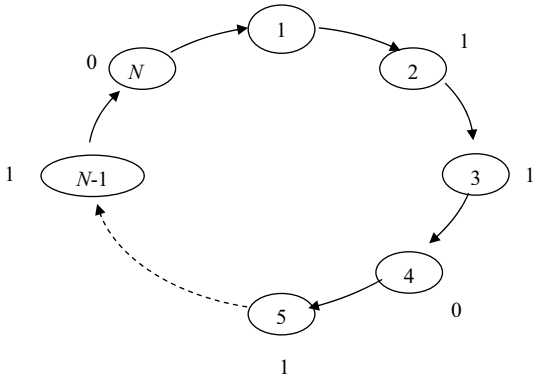


Рис. 4.2

Лема. Для того, щоб задача про сейф для контуру довжиною N мала p розв'язок, необхідно й достатньо, щоб

$$\sum_{i-1}^{N} b_i \equiv 0(\mathrm{mod}\,2). \tag{4.3}$$

Для доведення запишемо систему (4.1) щодо контуру на рис. 4.2.

Для цього сейфа система (4.1) набуде такого вигляду:

$$\left.\begin{array}{ccccccccccc} x_1 + & \cdot & & \cdot & & \cdot & \cdot & \cdot & & + x_N & \equiv b_1 \\ x_1 + & x_2 & & \cdot & & \cdot & \cdot & \cdot & & \cdot & \equiv b_2 \\ & x_2 + & x_3 & & \cdot & & \cdot & \cdot & & \cdot & \equiv b_3 \\ & & x_3 + & x_4 & & \cdot & \cdot & \cdot & & \cdot & \equiv b_4 \\ & & & & \cdot & & \cdot & \cdot & & \cdot & \cdot \\ & & & & & & & x_{N-1} & & + x_N & \equiv b_N \end{array}\right\} \mathrm{mod}\,2 \tag{4.4}$$

Якщо скласти всі рівності системи (4.4), то в лівій частині одержимо $0(\mathrm{mod}\,2)$, а в правій $-\sum_{i-1}^{N} b_i \equiv 0(\mathrm{mod}\,2)$, тобто результати додавання у лівій і

правій частинах збігаються. Це є необхідною умовою розв'язання системи (4.3). Доведемо її достатність. Для цього в системі (4.3) задамо $x_1 = b_1$. Тоді $x_2 = b_1 + b_2 (\mathrm{mod} 2)$, $x_3 = b_1 + b_2 + b_3 (\mathrm{mod} 2)$ і т. д. до $x_N = \sum_{i-1}^{N} b_i \equiv 0 (\mathrm{mod}\, 2)$, що повною мірою погоджується з першим рівнянням системи. Тим самим задача розв'язана. Аналогічно розв'язується задача про математичний сейф на мережах.

Розглянемо граф, що являє собою транспортну мережу, тобто орієнтований граф без контурів.

Теорема. Задача про сейф, заданий на транспортній мережі, завжди має розв'язок для довільного початкового стану.

Для доведення визначимо поняття рангу вершини. Надамо всім вершинам ранг $r(i) = 0$, у які не заходить жодна дуга. Ці вершини називаються джерелами мережі.

Якщо існує множина вершин рангу $p-1$ $(p \geq 1)$, і ще існують вершини з невизначеним рангом, то надамо ранг p усім джерелам мережі, що виникнуть після видалення вершин рангу $p-1$ разом з вихідними з них дугами. Вершини, з яких не виходить жодна дуга, називаються стоками мережі.

Алгоритм розв'язування задачі полягає в послідовному перемиканні тих замків (вершин), які перебувають у стані 1 у порядку зростання їхніх рангів. Черговість перемикання вершин однакового рангу довільна. Алгоритм закінчує роботу після того, як усі стоки мережі перейдуть у стан 0.

Наступна за складністю структура складається з декількох непересічних (незалежних) контурів, занурених у мережу. Мається на увазі, що якщо кожен контур стягти в одну вершину, то одержимо звичайну мережу без контурів. Варто врахувати, що кожен контур не повинен мати діагоналей у вигляді шляхів, тому що в цьому разі він представляв би об'єднання контурів, що мають загальні частини.

У процесі роботи описаного вище алгоритму на мережі вершини нульового рангу, потім першого й так далі послідовно переходять у нульовий стан. Усі

такі вершини разом з вихідними дугами можна видаляти, не впливаючи на розв'язок задачі. Розглянемо деякий контур C і виділимо множину вершин $\mu(C)$, що не належать цьому контуру, і з яких досяжна хоча б одна вершина цього контуру. Якщо в процесі деяких кроків усі вершини $\mu(C)$ переходять у нульовий стан, то стан контуру C, отриманий внаслідок цих операцій, назвемо канонічним.

Приклад 3. Розглянемо граф, заданий на рис. 4.3. У ньому N = 16 й існують три такі незалежні цикли: C_1 = (6.7.12), C_2 = (8, 9, 10, 11) і C_3 = (13, 14, 15). Для них

$$\mu(C_1) = (2, 5),\ \mu(C_2) = (1, 3, 4, 2, 5, 6, 7, 12),$$
$$\mu(C_3) = (1, 3, 4, 2, 5, 6, 7, 8, 9, 10, 11, 12).$$

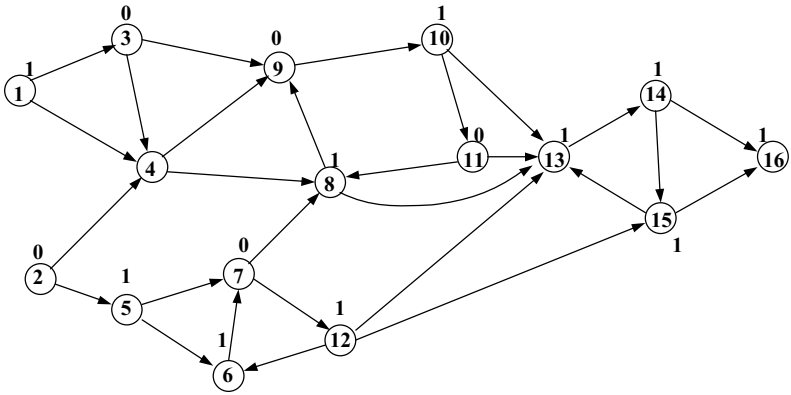


Рис. 4.3. Мережа з незалежними контурами

Переведемо вершину 5 у стан 0. Канонічний стан першого циклу буде таким: b_6 = 0, b_7 = 1, b_{12} = 1. Умова (4.3) виконується. Якщо тепер перевести вершину 7 у стан 0, то всі вершини першого циклу та вершина 8 перейдуть у той самий стан. Переведемо вершини 3 й 4 у стан 0. При цьому стан вершини 9 не зміниться, а вершина 8 перейде в стан 1. Канонічний стан другого циклу буде таким: b_8 = 1, b_9 = 0, b_{10} = 1 й b_{11} = 0.

Якщо тепер перевести вершину 10 у стан 0, то вершина 11 перейде в стан 1, а вершина 13 у стан 0. Далі переводимо вершину 11 (і 8 теж) у стан 0, а вершина 13 повернеться у стан 1. Канонічний стан третього циклу буде таким: $b_{13} = b_{14} = b_{15} = 1$. Це стан не задовольняє умову (4.3), тому вихідна задача нерозв'язна.

Як бачимо, у загальному випадку задача про математичний сейф на графах розв'язується досить простим шляхом. Але існують математичні сейфи іншої структури, які на графах задати досить складно. Такі сейфи задаються у вигляді матриць, де кожен елемент матриці відповідає початковому стану відповідного замка.

4.3. Однотипні математичні сейфи з простим числом станів замків

Розглянемо задачу про математичний сейф на матрицях. У всіх задачах про сейф на матрицях усі замки сейфа розташовані у вигляді прямокутної таблиці розміром $m \times n$, тобто у вигляді матриці $Z = (z_{ij})_{m,n}$. Для будь-якого замка z_{ij} вхідними замками вважаються замки, розташовані в тому самому рядку і в тому самому стовпці. Вихідний стан сейфа задається матрицею $B = (b_{ij})_{m,n}$. Нехай матриця $X = (x_{ij})_{m,n}$ — розв'язок задачі, де x_{ij} дорівнює числу поворотів ключа в замку z_{ij}.

Тоді умовою того, що елемент b_{ij} перетвориться матрицею X у нуль, буде співвідношення

$$\sum_{k=1}^{n} x_{ik} + \sum_{\substack{k=1 \\ k \neq i}}^{m} x_{kj} + b_{ij} \equiv 0 \,(\mathrm{mod}\, K), \tag{4.5}$$

де $i = 1, 2, \ldots, m$; $j = 1, 2, \ldots, n$.

Позначимо $\vec{x} = \left(x_{11}, x_{12}, \ldots, x_{1n}, x_{21}, x_{22}, \ldots, x_{2n}, \ldots, x_{m,n-1}, x_{mn}\right)$ вектор-стовпець, отриманий з матриці X послідовним записом її рядків.

Аналогічно з матриці B одержимо вектор-стовпець $\vec{b}$. Крім того, нехай $\Im_n$ — матриця розміру $n \times n$, що складається з одиниць, E_n — одинична матриця того самого розміру. Тоді співвідношення (4.5) для усієї матриці B запишеться у вигляді системи рівнянь

$$A\vec{x} + \vec{b} \equiv 0 (\mathrm{mod}\, K), \tag{4.6}$$

де матриця A розміру $mn \times mn$ складається з m^2 кліток:

$$A = \begin{pmatrix} \Im_n & E_n & E_n & .. & .. & E_n \\ E_n & \Im_n & E_n & .. & .. & E_n \\ E_n & E_n & \Im_n & .. & .. & E_n \\ .. & .. & .. & .. & .. & .. \\ E_n & E_n & E_n & .. & .. & \Im_n \end{pmatrix} \tag{4.7}$$

Деякі чисельні методи розв'язання таких систем розглядалися у [1].

Специфіка цієї задачі дозволяє знаходити розв'язок системи безпосередньо, тому що матриця A має стандартний вигляд і не залежить від значень матриці B. Її ранг і визначник залежать тільки від значень m та n.

Якщо ранг матриці A дорівнює mn, то розв'язок системи (4.6) має такий вигляд:

$$\vec{x} = -A^{-1}\vec{b} \,(\mathrm{mod}\, K). \tag{4.8}$$

Таким чином, проблема зводиться до відшукання оберненої матриці A^{-1}. У загальному випадку для довільних m, n вона може не існувати. Тоді система (4.6) може мати розв'язок, якщо початковий стан задовольняє певним обмеженням.

У роботі [2] була знайдена обернена матриця у вигляді так званої T-матриці.

$$T_{m,n}(\alpha_1, \alpha_2, \alpha_3, \alpha_4) = \begin{pmatrix} H_n(\alpha_1, \alpha_2) & H_n(\alpha_3, \alpha_4) & \ldots & H_n(\alpha_3, \alpha_4) \\ H_n(\alpha_3, \alpha_4) & H_n(\alpha_1, \alpha_2) & \ldots & H_n(\alpha_3, \alpha_4) \\ \ldots\ldots\ldots & \ldots\ldots\ldots & \ldots & \ldots\ldots\ldots \\ H_n(\alpha_3, \alpha_4) & H_n(\alpha_3, \alpha_4) & \ldots & H_n(\alpha_1, \alpha_2) \end{pmatrix}, \tag{4.9}$$

де

$$H_n(\alpha,\beta)=\begin{pmatrix} \alpha & \beta & \beta & \dots & \beta \\ \beta & \alpha & \beta & \dots & \beta \\ \beta & \beta & \alpha & \dots & \beta \\ \dots & \dots & \dots & \dots & \dots \\ \beta & \beta & \beta & \dots & \alpha \end{pmatrix}.$$

Шукатимемо обернену матрицю A^{-1} системи (4.6) у вигляді T-матриці $A^{-1}=T_{m,n}(\alpha_1,\alpha_2,\alpha_3,\alpha_4)$.

З огляду на те, що $AA^{-1}=E_{mn}$, там же [2] отримано розв'язок:

$$\left.\begin{aligned} \alpha_1 &\equiv \frac{1}{m-1}+\frac{1}{n-1}-1+\alpha_4 \\ \alpha_2 &\equiv \frac{1}{n-1}+\alpha_4 \\ \alpha_3 &\equiv \frac{1}{m-1}+\alpha_4 \\ \alpha_4 &\equiv -\left(\frac{1}{n-1}+\frac{1}{m-1}\right)\frac{1}{m+n-1} \end{aligned}\right\} (\mathrm{mod}\, K). \qquad (4.10)$$

Введемо позначення:

$$-\sum_{j=1}^{n} b_{ij}=\lambda_i,\ i=1,2,...,m,$$

$$-\sum_{i=1}^{m} b_{ij}=\beta_j,\ j=1,2,...,n.$$

Якщо $m\neq 1(\mathrm{mod}\, K)$, $n\neq 1(\mathrm{mod}\, K)$, $m+n\neq 1(\mathrm{mod}\, K)$, то підставляючи значення α_i у формулу (4.8), отримаємо:

$$x_{ij}=-b_{ij}\frac{1}{m-1}-b_{ij}\frac{1}{n-1}+b_{ij}-b_{ij}\alpha_4+\frac{1}{n-1}\lambda_i+\alpha_4\lambda_i+b_{ij}\frac{1}{n-1}+b_{ij}\alpha_4+b_j\frac{1}{m-1}+\alpha_4 b_j+$$
$$+b_{ij}\frac{1}{m-1}+b_{ij}\alpha_4+\alpha_4\sum_{k=1}^{m}\lambda_k-\alpha_4\lambda_i-\alpha_4 b_j-\alpha_4 b_{ij}=b_{ij}+\frac{1}{n-1}\lambda_i+\frac{1}{m-1}b_j+\alpha_4\sum_{k=1}^{m}\lambda_k.$$

Після спрощень остаточно отримаємо:

$$x_{ij} = b_{ij} + \frac{1}{n-1}\lambda_i + \frac{1}{m-1}b_j + \left(\frac{1}{n-1} + \frac{1}{m-1}\right)\frac{1}{m+n-1}\sum_{k=1}^{m}\lambda_k . \qquad (4.11)$$

Розглянемо математичні сейфи з однотипними замками, у яких число станів *K* — просте число. Для розв'язку задачі введемо позначення: $S_i = x_{i1} + x_{i2} + ... + x_{in}$, $S = \sum_{i=1}^{m} S_i$, $\Sigma_j = x_{1j} + x_{2j} + ... + x_{nj}$. і скористаємося методом виділення підсистем, суть якого полягає в наступному.

Утворимо підсистему із системи (4.6) шляхом додавання перших *n* рівнянь, других *n* рівнянь і т. д. У результаті отримаємо таку підсистему з *m* рівнянь:

$$\begin{aligned} nS_1 + S_2 + S_3 + ... + S_{m-1} + S_m &= \lambda_1 (\bmod K) \\ S_1 + nS_2 + S_3 + ... + S_{m-1} + S_m &= \lambda_2 (\bmod K) \\ &\ldots \\ S_1 + S_2 + S_3 + ... + S_{m-1} + nS_m &= \lambda_m (\bmod K) \end{aligned} \qquad (4.12)$$

Назвемо її підсистемою першого роду. Якщо додати в цій підсистемі всі рівняння, то отримаємо рівність

$$(m+n-1)S = \sum_{i=1}^{m} \lambda_i . \qquad (4.13)$$

Підсистему другого роду отримаємо із системи (4.6) у такий спосіб. Виділимо (*kn*+*j*)-і рівняння, де $k = 0, 1, 2, ..., m-1$. Отримаємо систему з *m* рівнянь

$$\begin{aligned} S_1 + x_{2j} + x_{3j} + ... + x_{nj} &= -b_{1j} (\bmod K) \\ x_{1j} + S_2 + x_{3j} + ... + x_{nj} &= -b_{2j} (\bmod K) \\ &\ldots \\ x_{1j} + x_{2j} + x_{3j} + ... + S_m &= -b_{mj} (\bmod K). \end{aligned} \qquad (4.14)$$

Додаючи ці рівняння, отримаємо рівність

$$S + (m-1)\Sigma_j = b_j , j = 1,2,...,n . \qquad (4.15)$$

Залежно від значень m і n отримаємо чотири різні випадки.

Випадок 1. $m \neq 1 (\mathrm{mod}\, K)$, $n \neq 1 (\mathrm{mod}\, K)$, $m + n \neq 1 (\mathrm{mod}\, K)$. У цьому випадку з рівності (4.13) знаходимо $S = \sum_{i=1}^{m} \frac{\lambda_i}{m+n-1}$. Підставляючи цей вираз у рівність (4.15), отримаємо

$$\Sigma_j = \frac{1}{m-1}\left(b_j - S\right),\ j = 1,2,...,n. \tag{4.16}$$

З i-го рівняння системи (4.8) знаходимо

$$S_i = \frac{1}{n-1}\left(\lambda_i - S\right),\ i = 1,2,...,m. \tag{4.17}$$

Додамо в ліву та праву частини *i-го* рівняння підсистеми (4.14) $x_{ij.}$ Отримаємо рівність $S_i + \Sigma_j = -b_{ij} + x_{ij}$. Звідси

$$x_{ij} = b_{ij} + S_i + \Sigma_j. \tag{4.18}$$

Підставляючи сюди вирази (4.16) і (4.17), отримаємо розв'язок системи (4.6) $x_{ij} = b_{ij} + \frac{1}{n-1}\lambda_i + \frac{1}{m-1}b_j + \left(\frac{1}{n-1} + \frac{1}{m-1}\right)\frac{1}{m+n-1}\sum_{k=1}^{m}\lambda_k$, що збігається з (4.11), отриманим раніше іншим способом.

Випадок 2. $m \neq 1 (\mathrm{mod}\, K)$, $n \neq 1 (\mathrm{mod}\, K)$, $m + n = 1 (\mathrm{mod}\, K)$.

Якщо S відомо, то розв'язок $X = \left(x_{ij}\right)_{m,n}$ знаходимо так само, як й у випадку 1. Тому, задаючи значення S, отримаємо новий розв'язок. Якщо до матриці X додати матрицю $Y = (y_{ij} = k)_{mn}$, то розв'язок не зміниться. Це випливає з того, що на елемент b_{ij} у матриці Y діє величина, рівна $(m + n - 1)k = 0$, тобто $X + Y$ є також розв'язком. Таким чином, отримаємо ще додаткові K–1 розв'язки, а всього — K розв'язків. Це випливає також з того, що можна довільним чином задавати K значень S.

Приклад 1. Нехай $B = \begin{pmatrix} 3 & 1 & 0 & 0 \\ 0 & 0 & 2 & 4 \end{pmatrix}$, K = 5 й S = 1. Переконаємося, що виконується попередня умова (4.13), а саме, $\sum_{i=1}^{2}\lambda_i = -4 - 6 = 0 (\mathrm{mod}\, 5)$, що й потрібно.

Тоді за формулами (4.16), (4.17) знаходимо $S_1 = 0, S_2 = 1,$ $\Sigma_1 = 1, \Sigma_2 = 3, \Sigma_3 = 2, \Sigma_4 = 0.$ Підставляючи ці значення в (4.18), отримаємо розв'язок системи (4.6) $X = \begin{pmatrix} -1 & -1 & 2 & 0 \\ 2 & -1 & 0 & 0 \end{pmatrix}$.

Перевіримо цей розв'язок.

$$B = \begin{pmatrix} 3 & 1 & 0 & 0 \\ 0 & 0 & 2 & 4 \end{pmatrix} \xrightarrow{x_{11} = -1} \begin{pmatrix} 2 & 0 & -1 & -1 \\ -1 & 0 & 2 & 4 \end{pmatrix} \xrightarrow{x_{12} = -1} \begin{pmatrix} 1 & -1 & -2 & -2 \\ -1 & -1 & 2 & 4 \end{pmatrix} \xrightarrow{x_{13} = 2} \begin{pmatrix} 3 & 1 & 0 & 0 \\ -1 & -1 & -1 & 4 \end{pmatrix} \xrightarrow{x_{21} = 2}$$

$$\xrightarrow{x_{22} = -1} \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix} \to \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Додамо до цього розв'язку $Y = \begin{pmatrix} 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 \end{pmatrix}$. У результаті отримаємо $X_1 = X + Y = \begin{pmatrix} 1 & 1 & -1 & 2 \\ -1 & 1 & 2 & 2 \end{pmatrix}$. Це відповідає початковому значенню $S = 2, S_1 = 3, S_2 = -1, \Sigma_1 = 0, \Sigma_2 = 2, \Sigma_3 = 1, \Sigma_4 = -1.$ Неважко переконатися, що X_1 є теж розв'язком.

Випадок 3. $m = 1 (\mathrm{mod}\, K),\ n \neq 1 (\mathrm{mod}\, K).$ З рівності (4.15) випливає, що $S = b_j$. Значить усі $b_j = \beta, j = 1,2,...,n$. Це є першою попередньою умовою розв'язку задачі. Другою попередньою умовою повинно бути $\beta = \sum_{i=1}^{m} \frac{\lambda_i}{m + n - 1}$.

Знаходимо значення S_i з рівності (4.17), а із системи (4.14), послідовно віднімаючи з першого i – і рівняння, знаходимо

$$x_{ij} = b_{ij} - b_{1j} + S_i - S_1 + x_{1j}, i = 2,3,...,m, j = 1,2,...,n. \qquad (4.19)$$

Задаючи довільно значення x_{1j} таким чином, щоб $\sum_{j=1}^{n} x_{1j} = S_1$, отримаємо розв'язок $X = (x_{ij})_{mn}$. Додамо до цього розв'язку матрицю

$Y = \begin{pmatrix} y_1 & y_2 & \dots & y_n \\ y_1 & y_2 & \dots & y_n \\ \dots & \dots & \dots & \dots \\ y_1 & y_2 & \dots & y_n \end{pmatrix}$, де $\sum_{j=1}^{n} y_j = 0 \,(\mathrm{mod} K)$. При цьому матриця Y змінить елемент b_{ij} на величину $(m-1)y_j + \sum_{k=1}^{n} y_k = 0 \,(\mathrm{mod} K)$. Отже $X + Y$ дає новий розв'язок задачі. Усього таких розв'язків буде K^{n-1}, тобто дорівнює числу розв'язків рівняння $\sum_{j=1}^{n} y_j = 0 \,(\mathrm{mod} K)$.

Існує ще випадок 4, для якого $m \neq 1(\mathrm{mod}\, K)$, $n = 1(\mathrm{mod}\, K)$. У цьому випадку розв'язок досягається природним шляхом, і він не єдиний.

4.4. Математичні сейфи з довільною кількістю станів замків

Якщо K — не просте число, то незважаючи на виконання необхідних умов для чисел m і n, можуть виникати ситуації, коли задача не має розв'язку. У такому разі треба знайти необхідну корекцію початкового стану сейфа, щоб задача мала розв'язок. Розглянемо цю задачу при виконанні таких основних умов:

$$m + n \neq 1 \,(\mathrm{mod}\, K),\ n \neq 1 \,(\mathrm{mod}\, K),\ m \neq 1 (\mathrm{mod}\, K).$$

Введемо позначення $\sum_{\lambda=1}^{m} \sum_{\eta=1}^{n} b_{\lambda\eta} = \sum (b)$; $\sum_{\lambda=1}^{m} b_{\lambda j} = \lambda_j$; $\sum_{\eta=1}^{n} b_{i\eta} = \sigma_i$.

Під час розв'язання такої задачі виникають п'ять таких випадків:

а) $НОД(m+n-1, K) = d > 1$, $НОД(m-1, K) = НОД(n-1, K) = 1$;

б) $НОД(n-1, K) = d > 1$, $НОД(m-1, K) = НОД(m+n-1, K) = 1$;

в) $НОД(m+n-1, K) = d_1 > 1$, $НОД(n-1, K) = d_2$, $НОД(m-1, K) = 1$;

г) $НОД(m+n-1, K) = d_1 > 1$, $НОД(m-1, K) = d_2 > 1$, $НОД(n-1, K) = 1$;

д) $НОД(m-1, K) = d_1 > 1$, $НОД(n-1, K) = d_2 > 1$, $НОД(m+n-1, K) = 1$.

Розглянемо розв'язок цієї задачі для випадку **а**). Щоб доданок у (4.6), що має в знаменнику число $m+n-1$ можна було перетворити в ціле число, необхідно розв'язати рівняння

$$t(m+n-1) \equiv \sum(b) (\mathrm{mod}\, K). \tag{4.20}$$

Однак при умовах **а**) і для довільних b_{ij} це не завжди можливо. Розв'язок може існувати тільки при обмеженні

$$\sum(b) \equiv 0 (\mathrm{mod}\, d). \tag{4.21}$$

Тоді (4.20) матиме d розв'язків виду

$$t = \left[t_1 + \frac{K}{d}(s-1) \right] (\mathrm{mod}\, K), \tag{4.22}$$

де $s = 1, 2, \ldots, d$, а t_1 — розв'язок рівняння

$$t_1 \left(\frac{m+n-1}{d} \right) \equiv \frac{\sum(b)}{d} \left(\mathrm{mod} \frac{K}{d} \right); \tag{4.23}$$

Задача має d розв'язків і запишеться у вигляді

$$x_{ij} \equiv \left[b_{ij} - \frac{\sigma_i}{n-1} - \frac{\lambda_j}{m-1} + \left(\frac{1}{m-1} + \frac{1}{n-1} \right) t_1 + \frac{K}{d}(s-1) \right] (\mathrm{mod}\, K), \tag{4.24}$$

де $s = 1, 2, \ldots, d$.

Приклад 1. Нехай $K = 9$, $m = 2$, $n = 5$. Початковий стан сейфа задано матрицею B. Параметри задовольняють основним умовам, а $НОД(m+n-1, K) = НОД(6, 9) = 3 > 1$. Сума всіх елементів матриці B дорівнює $41 \neq 0 (\mathrm{mod}\, 3)$. Введемо корекцію та отримаємо B' (елемент корекції виділений рамкою).

$$B = \begin{pmatrix} 7 & 5 & 5 & 5 & 5 \\ 2 & 2 & \boxed{4} & 2 & 4 \end{pmatrix}, \quad B' = \begin{pmatrix} 7 & 5 & 5 & 5 & 5 \\ 2 & 2 & 4 & 2 & 5 \end{pmatrix}.$$

Обчислимо необхідні параметри:

$$\sigma_1 = 0; \sigma_2 = 6; \lambda_1 = \lambda_3 = 0; \lambda_2 = \lambda_4 = 7; \lambda_5 = 1;$$

$$-\frac{1}{m-1}\equiv -1\ (\mathrm{mod}\,9);\quad -\frac{1}{n-1}\equiv\frac{1}{4}\equiv -2\ (\mathrm{mod}\,9);$$

$$t_1\equiv\frac{\left(\frac{42}{3}\right)}{\left(\frac{6}{3}\right)}(\mathrm{mod}\,3)\equiv 1(\mathrm{mod}\,3).$$

$$x_{ij}=\left[b_{ij}+2\sigma_i-\lambda_j-1+3(s-1)\right];\quad s=1,2,3.$$

Після підстановки всіх параметрів отримаємо 3 розв'язки задачі:

$$X^{(1)}=\begin{pmatrix}6&6&4&6&3\\4&6&6&6&6\end{pmatrix},\quad X^{(2)}=\begin{pmatrix}0&0&7&0&6\\7&0&0&0&0\end{pmatrix},\quad X^{(3)}=\begin{pmatrix}3&3&1&3&0\\1&3&3&3&3\end{pmatrix}.$$

Перевіримо розв'язки $X^{(2)}$ і $X^{(3)}$.

$$X^{(2)}:\left(\begin{array}{c|c|c|c|c}7&5&5&5&5\\2&2&4&2&5\end{array}\right)\xrightarrow{+7}\left(\begin{array}{c|c|c|c|c}5&3&3&3&3\\2&2&2&2&5\end{array}\right)\xrightarrow{+6}\left(\begin{array}{c|c|c|c|c}2&0&0&0&0\\2&2&2&2&2\end{array}\right)\xrightarrow{+7}\begin{pmatrix}0&0&0&0&0\\0&0&0&0&0\end{pmatrix}.$$

$$X^{(3)}:\left(\begin{array}{c|c|c|c|c}7&5&5&5&5\\2&2&4&2&5\end{array}\right)\xrightarrow{+3}\left(\begin{array}{c|c|c|c|c}1&-1&-1&-1&-1\\5&2&4&2&5\end{array}\right)\xrightarrow{+3}\left(\begin{array}{c|c|c|c|c}4&2&2&2&2\\5&5&4&2&5\end{array}\right)\xrightarrow{+1}$$

$$\xrightarrow{+3}\left(\begin{array}{c|c|c|c|c}5&3&3&3&3\\5&5&5&2&5\end{array}\right)\xrightarrow{+1}\left(\begin{array}{c|c|c|c|c}-1&6&6&6&6\\5&5&5&5&5\end{array}\right)\xrightarrow{+3}\left(\begin{array}{c|c|c|c|c}0&6&6&6&6\\6&6&6&6&6\end{array}\right)\xrightarrow{+3}$$

$$\xrightarrow{+3}\left(\begin{array}{cc|c|c|c}0&0&6&6&6\\0&0&0&0&0\end{array}\right)\xrightarrow{+3}\left(\begin{array}{ccc|c|c}0&0&0&6&6\\3&3&3&3&3\end{array}\right)\xrightarrow{+3}\left(\begin{array}{ccccc}0&0&0&0&6\\6&6&6&6&6\end{array}\right)\xrightarrow{+3}\begin{pmatrix}0&0&0&0&0\\0&0&0&0&0\end{pmatrix}.$$

Всі інші нерозглянуті випадки розв'язуються за подібною схемою.

4.5. Математичні сейфи з різними типами замків

Розглянемо таку задачу на прикладі математичного сейфа з двома типами замків. Це відповідає тому, що в загальній системи (4.5) перші p рядків матриці B представляють замки першого типу із числом станів k_1, а інші рядки

— замки другого типу з числом станів k_2. Це приведе до того, що система загального виду (4.5) задаватиметься співвідношеннями виду:

$$\sum_{k=1}^{n} x_{ik} + \sum_{\substack{k=1 \\ k\neq i}}^{m} x_{kj} + b_{ij} \equiv 0 (\mathrm{mod}\, k_1), (i = 1, 2, \ldots, p); \tag{4.25}$$

$$\sum_{k=1}^{n} x_{ik} + \sum_{\substack{k=1 \\ k\neq i}}^{m} x_{kj} + b_{ij} \equiv 0 (\mathrm{mod}\, k_2), (i = p+1, p+2, \ldots, m). \tag{4.26}$$

Якщо помножити (4.25) на k_2, а (4.26) — на k_1, то співвідношення не зміняться, але тепер їх можна записати у вигляді

$$\boldsymbol{A}' \vec{x} + \boldsymbol{b}' \equiv 0 (\mathrm{mod}\, k_1 k_2), \tag{4.27}$$

де $\boldsymbol{A}'$ — та сама матриця з (4.1), у якої pm перші рядки помножені на k_2, а інші — на k_1, тобто

$$\boldsymbol{A}' = \begin{pmatrix} k_2\mathfrak{I}_n & k_2E_n & k_2E_n & .... & k_2E_n \\ k_2E_n & k_2\mathfrak{I}_n & k_2E_n & .... & k_2E_n \\ ... & .... & .... & .... & .... \\ k_1E_n & k_1E_n & k_1E_n & .... & k_1E_n \\ .... & .... & .... & .... & .... \\ k_1E_n & k_1E_n & k_1E_n & .... & k_1\mathfrak{I}_n \end{pmatrix}, \tag{4.28}$$

а $\boldsymbol{b}' = (k_2 b_1, k_2 b_2, .. k_2 b_{pn}, k_1 b_{pn+1}, k_1 b_{pn+2} ..., k_1 b_{mn})$.

Неважко переконатись, що розв'язок системи (4.27) задовольняє систему рівнянь

$$A\boldsymbol{x} + \boldsymbol{b} \equiv 0 (\mathrm{mod}\, k_1 k_2),$$

де A — матриця (4.7).

Задачу про математичні сейфи на матрицях із двома типами замків, тобто систему (4.27), можна розв'язувати трьома методами: а) методом, що використовує T-матриці, б) методом Крамера, в) методом виділення підсистем, описаного в [3]. Розглянемо приклад розв'язання задачі методом (а).

Приклад 1. Нехай $m = 5,\ n = 3,\ k_1 = 3,\ k_2 = 5,\ p = 2$, а матриця B має вигляд

$$B = \begin{pmatrix} 1 & 2 & 0 \\ 1 & 1 & 1 \\ 4 & 1 & 0 \\ 4 & 2 & 1 \\ 0 & 2 & 0 \end{pmatrix}.$$

Обчислимо дробові значення в (4.10):

$$\begin{cases} \dfrac{1}{m-1} = \dfrac{1}{4} \equiv 4 (\mathrm{mod}\, 3 \cdot 5), \\ \dfrac{1}{n-1} = \dfrac{1}{2} \equiv 8 (\mathrm{mod}\, 3 \cdot 5), \\ \dfrac{1}{m+n-1} = \dfrac{1}{7} \equiv 13 (\mathrm{mod}\, 3 \cdot 5). \end{cases}$$

Підставляючи ці значення в (4.10), одержуємо: $\alpha_1 = 5,\ \alpha_2 = 2,\ \alpha_3 = 13,\ \alpha_4 = 9$. Звідси одержуємо обернену матрицю (4.7) $A^{-1} = T_{5,3}(5,2,13,9)$ тобто

$$A^{-1} = \begin{pmatrix}
5 & 2 & 2 & 13 & 9 & 9 & 13 & 9 & 9 & 13 & 9 & 9 & 13 & 9 & 9 \\
2 & 5 & 2 & 9 & 13 & 9 & 9 & 13 & 9 & 9 & 13 & 9 & 9 & 13 & 9 \\
2 & 2 & 5 & 9 & 9 & 13 & 9 & 9 & 13 & 9 & 9 & 13 & 9 & 9 & 13 \\
13 & 9 & 9 & 5 & 2 & 2 & 13 & 9 & 9 & 13 & 9 & 9 & 13 & 9 & 9 \\
9 & 13 & 9 & 2 & 5 & 2 & 9 & 13 & 9 & 9 & 13 & 9 & 9 & 13 & 9 \\
9 & 9 & 13 & 2 & 2 & 5 & 9 & 9 & 13 & 9 & 9 & 13 & 9 & 9 & 13 \\
13 & 9 & 9 & 13 & 9 & 9 & 5 & 2 & 2 & 13 & 9 & 9 & 13 & 9 & 9 \\
9 & 13 & 9 & 9 & 13 & 9 & 2 & 5 & 2 & 9 & 13 & 9 & 9 & 13 & 9 \\
9 & 9 & 13 & 9 & 9 & 13 & 2 & 2 & 5 & 9 & 9 & 13 & 9 & 9 & 13 \\
13 & 9 & 9 & 13 & 9 & 9 & 13 & 9 & 9 & 5 & 2 & 2 & 13 & 9 & 9 \\
9 & 13 & 9 & 9 & 13 & 9 & 9 & 13 & 9 & 2 & 5 & 2 & 9 & 13 & 9 \\
9 & 9 & 13 & 9 & 9 & 13 & 9 & 9 & 13 & 2 & 2 & 5 & 9 & 9 & 13 \\
13 & 9 & 9 & 13 & 9 & 9 & 13 & 9 & 9 & 13 & 9 & 9 & 5 & 2 & 2 \\
9 & 13 & 9 & 9 & 13 & 9 & 9 & 13 & 9 & 9 & 13 & 9 & 2 & 5 & 2 \\
9 & 9 & 13 & 9 & 9 & 13 & 9 & 9 & 13 & 9 & 9 & 13 & 2 & 2 & 5
\end{pmatrix}$$

Нескладно підрахувати, що $\vec{x} = -A^{-1}\,\boldsymbol{b} = (12, 6, 13, 12, 5, 14, 14, 4, 12, 13, 4, 12, 4, 14, 6)$, або

$$X = \begin{pmatrix} 12 & 6 & 13 \\ 12 & 5 & 14 \\ 14 & 4 & 12 \\ 13 & 4 & 12 \\ 4 & 14 & 6 \end{pmatrix} (\mathrm{mod}\, k_1 k_2\ ).$$

Можна безпосередньо переконатися, що це є розв'язок (4.27).

Перевіримо цей розв'язок, вказуючи над матрицею кількість поворотів відповідного ключа.

$$B = \overset{x_{11}=12}{\begin{pmatrix} 1 & 2 & 0 \\ 1 & 1 & 1 \\ 4 & 1 & 0 \\ 4 & 2 & 1 \\ 0 & 2 & 0 \end{pmatrix}} \to \overset{x_{12}=6}{\begin{pmatrix} 13 & 14 & 12 \\ 13 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 2 & 1 \\ 12 & 2 & 0 \end{pmatrix}} \to \overset{x_{13}=13}{\begin{pmatrix} 4 & 5 & 3 \\ 13 & 7 & 1 \\ 1 & 7 & 0 \\ 1 & 8 & 1 \\ 12 & 8 & 0 \end{pmatrix}} \to \overset{x_{21}=12}{\begin{pmatrix} 2 & 3 & 1 \\ 13 & 7 & 14 \\ 1 & 7 & 13 \\ 1 & 8 & 14 \\ 12 & 8 & 13 \end{pmatrix}} \to$$

$$\to \overset{x_{22}=5}{\begin{pmatrix} 14 & 3 & 1 \\ 10 & 4 & 12 \\ 13 & 7 & 13 \\ 13 & 8 & 14 \\ 9 & 8 & 13 \end{pmatrix}} \to \overset{x_{23}=14}{\begin{pmatrix} 14 & 8 & 1 \\ 0 & 9 & 1 \\ 13 & 12 & 13 \\ 13 & 13 & 14 \\ 9 & 13 & 13 \end{pmatrix}} \to \overset{x_{31}=14}{\begin{pmatrix} 14 & 8 & 0 \\ 14 & 8 & 0 \\ 13 & 12 & 12 \\ 13 & 13 & 13 \\ 9 & 13 & 12 \end{pmatrix}} \to \overset{x_{32}=4}{\begin{pmatrix} 13 & 8 & 0 \\ 13 & 8 & 0 \\ 12 & 11 & 11 \\ 12 & 13 & 13 \\ 8 & 13 & 12 \end{pmatrix}} \to$$

$$\to \overset{x_{33}=12}{\begin{pmatrix} 13 & 12 & 0 \\ 13 & 12 & 0 \\ 1 & 0 & 0 \\ 12 & 2 & 13 \\ 8 & 2 & 12 \end{pmatrix}} \to \overset{x_{41}=13}{\begin{pmatrix} 13 & 12 & 12 \\ 13 & 12 & 12 \\ 13 & 12 & 12 \\ 12 & 2 & 10 \\ 8 & 2 & 9 \end{pmatrix}} \to \overset{x_{42}=4}{\begin{pmatrix} 11 & 12 & 12 \\ 11 & 12 & 12 \\ 11 & 12 & 12 \\ 10 & 0 & 8 \\ 6 & 2 & 9 \end{pmatrix}} \to \overset{x_{43}=12}{\begin{pmatrix} 11 & 1 & 12 \\ 11 & 1 & 12 \\ 11 & 1 & 12 \\ 14 & 4 & 12 \\ 6 & 6 & 9 \end{pmatrix}} \to$$

$$\to \overset{x_{51}=4}{\begin{pmatrix} 11 & 1 & 9 \\ 11 & 1 & 9 \\ 11 & 1 & 9 \\ 11 & 1 & 9 \\ 6 & 6 & 6 \end{pmatrix}} \to \overset{x_{52}=16}{\begin{pmatrix} 0 & 1 & 9 \\ 0 & 1 & 9 \\ 0 & 1 & 9 \\ 0 & 1 & 9 \\ 10 & 10 & 10 \end{pmatrix}} \to \overset{x_{53}=6}{\begin{pmatrix} 0 & 0 & 9 \\ 0 & 0 & 9 \\ 0 & 0 & 9 \\ 0 & 0 & 9 \\ 9 & 9 & 9 \end{pmatrix}} \to \left.\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}\right\} (\mathrm{mod}\, 15) = B_{fin}.$$

Під час застосування цього методу виникають певні труднощі, якщо в знаменниках для α_i з'являються дільники нуля. Наприклад, якщо матриця B має розмірність 4×6, то в знаменниках при α_i з'являються дроби $\frac{1}{3}$ і $\frac{1}{5}$, яким немає відповідності в цілих числах для $k_1 = 3$ і $k_2 = 5$. Якщо представити ці дроби уявними числами, то можна домогтися їхнього скорочення тільки завдяки специфічному виду вектора $\boldsymbol{b}$.

Щодо рішення основної системи методом (б) можна сказати, що він пов'язаний з обчисленням великої кількості визначників величезних розмірів, що робить цей метод практично непридатним.

Перейдемо до викладу методу (в). Для цього з матриці (4.28) виділимо два роди підсистем. Підсистему першого роду отримаємо, якщо просумувати послідовно кожні n рівнянь, тобто підсистему виду

$$\begin{aligned}
&k_2(nS_1 + S_2 + S_3 + ... + S_{m-1} + S_m) \equiv k_2\lambda_1 (\mathrm{mod}\, k_1k_2) \\
&\qquad \text{……......} \\
&k_2(S_1 + S_2 + S_3 + ... + S_{m-1} + nS_m) \equiv k_2\lambda_p (\mathrm{mod}\, k_1k_2) \\
&k_1(nS_1 + S_2 + S_3 + ... + S_{m-1} + S_m) \equiv k_1\lambda_{p+1} (\mathrm{mod}\, k_1k_2) \qquad (4.29) \\
&\qquad \text{……......} \\
&k_1(S_1 + S_2 + S_3 + ... + S_{m-1} + nS_m) \equiv k_1\lambda_m (\mathrm{mod}\, k_1k_2),
\end{aligned}$$

де $S_i = x_{i1} + x_{i2} + ... + x_{in}$, $\lambda_i = -\sum_{j=1}^{n} b_{ij}$, $i = 1,2,...,m$, $j = 1,2,...,n$.

Підсистему другого роду для кожного $j = 1,2,...,n$, визначимо з тієї самої матриці в такий спосіб. Виділимо (kn+j)-і рівняння, де k = 0, 1, 2,……, m–1. Отримаємо для кожного $j = 1,2,...,n$, систему з m рівнянь:

$$\begin{aligned}
&k_2(S_1 + x_{2j} + x_{3j} + ... + x_{nj}) \equiv -k_2b_{1j} (\mathrm{mod}\, k_1k_2) \\
&\qquad \text{……} \\
&k_2(x_{1j} + S_2 + x_{3j} + ... + x_{nj}) \equiv -k_2b_{pj} (\mathrm{mod}\, k_1k_2)
\end{aligned}$$

$$k_1(x_{1j} + S_2 + x_{3j} + ... + x_{nj}) \equiv -k_1 b_{p+1j} (\bmod k_1 k_2) \quad (4.30)$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$k_1(x_{1j} + x_{2j} + x_{3j} + ... + S_m) \equiv -k_1 b_{mj} (\bmod k_1 k_2).$$

Опишемо алгоритм *p* розв’язку задачі.

Етап 1. Виконується за такою схемою.

Крок 1. З перших p рівнянь знаходимо вираження S_i, $i = 2,..., p$ через S_1.

Крок 2. У рівняннях, що залишилися, знаходимо вираження S_i, $i > p+1$, через S_{p+1}.

Крок 3. Підставляємо отримані вираження в перше рівняння і знаходимо визначення S_{p+1} через S_1.

Крок 4. Підставляємо всі значення S_i, $i = 2,...,m$ в останнє рівняння та знаходимо значення S_1, а потім усі значення S_i.

Етап 2. Для кожної j-ї підсистеми другого роду знаходимо значення змінних $x_{1j}, x_{2j},..., x_{mj}$ за схемою, описаної в етапі 1), але відносно змінних x_{1j}, x_{p+1j}. У результаті отримаємо матрицю X, тобто розв’язок задачі.

Приклад 2. Нехай $m = 3$, $n = 3$, $k_1 = 3$, $k_2 = 5$, $p = 2$, а матриця *B* має вигляд

$$B = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -4 \end{pmatrix}.$$

Переконаємося, що цю задачу можна розв’язати методом (а). Спочатку знайдемо обернену матрицю у вигляді *T*-матриці. Знаходимо значення α_i за модулем 15.

$$\alpha_4 = -\left(\frac{1}{2} + \frac{1}{2}\right)\left(\frac{1}{3+3-1}\right) = -\frac{1}{5},\ \alpha_1 = \frac{1}{2} + \frac{1}{2} - 1 + \alpha_4 = -\frac{1}{5},\ \alpha_2 = \alpha_3 = \frac{1}{2} + \alpha_4 = \frac{3}{10}.$$

Підставивши ці значення у обернену матрицю, отримаємо

$$A^{-1}=\begin{pmatrix}
-\frac{1}{5} & \frac{3}{10} & \frac{3}{10} & \frac{3}{10} & -\frac{1}{5} & -\frac{1}{5} & \frac{3}{10} & -\frac{1}{5} & -\frac{1}{5} \\
\frac{3}{10} & -\frac{1}{5} & \frac{3}{10} & -\frac{1}{5} & \frac{3}{10} & -\frac{1}{5} & -\frac{1}{5} & \frac{3}{10} & -\frac{1}{5} \\
\frac{3}{10} & \frac{3}{10} & -\frac{1}{5} & -\frac{1}{5} & -\frac{1}{5} & \frac{3}{10} & -\frac{1}{5} & -\frac{1}{5} & \frac{3}{10} \\
\frac{3}{10} & -\frac{1}{5} & -\frac{1}{5} & -\frac{1}{5} & \frac{3}{10} & \frac{3}{10} & \frac{3}{10} & -\frac{1}{5} & -\frac{1}{5} \\
-\frac{1}{5} & \frac{3}{10} & -\frac{1}{5} & \frac{3}{10} & -\frac{1}{5} & \frac{3}{10} & -\frac{1}{5} & \frac{3}{10} & -\frac{1}{5} \\
-\frac{1}{5} & -\frac{1}{5} & \frac{3}{10} & \frac{3}{10} & \frac{3}{10} & -\frac{1}{5} & -\frac{1}{5} & -\frac{1}{5} & \frac{3}{10} \\
\frac{3}{10} & -\frac{1}{5} & -\frac{1}{5} & \frac{3}{10} & -\frac{1}{5} & -\frac{1}{5} & -\frac{1}{5} & \frac{3}{10} & \frac{3}{10} \\
-\frac{1}{5} & \frac{3}{10} & -\frac{1}{5} & -\frac{1}{5} & \frac{3}{10} & -\frac{1}{5} & \frac{3}{10} & -\frac{1}{5} & \frac{3}{10} \\
-\frac{1}{5} & -\frac{1}{5} & \frac{3}{10} & -\frac{1}{5} & -\frac{1}{5} & \frac{3}{10} & \frac{3}{10} & \frac{3}{10} & -\frac{1}{5}
\end{pmatrix}$$

Множачи цю матрицю на вектор ***b***, отримаємо такі значення вектора $\boldsymbol{x}=(-1,7,9,7,-1,1,9,1,-1)$, що відповідає матриці $X=\begin{pmatrix}-1 & 7 & 9\\ 7 & -1 & 1\\ 9 & 1 & -1\end{pmatrix}$.

Тут, завдяки вдалому виду вектора ***b***, отримані цілі значення змінних.

Перевіримо цей розв'язок.

$$B=\begin{pmatrix}-1 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & -4\end{pmatrix}\xrightarrow{x_{11}=-1}\begin{pmatrix}-2 & -1 & -1\\ -1 & 0 & 0\\ -1 & 0 & -4\end{pmatrix}\xrightarrow{x_{12}=7}\begin{pmatrix}5 & 6 & 6\\ -1 & 7 & 0\\ -1 & 7 & -4\end{pmatrix}\xrightarrow{x_{13}=9}\begin{pmatrix}-1 & 0 & 0\\ -1 & 7 & 9\\ -1 & 7 & 5\end{pmatrix}\xrightarrow{x_{21}=7}$$

$$\xrightarrow{x_{22}=-1}\begin{pmatrix}6 & 0 & 0\\ 6 & -1 & 1\\ 6 & 7 & 5\end{pmatrix}\xrightarrow{x_{23}=1}\begin{pmatrix}6 & -1 & 0\\ 5 & -2 & 0\\ 6 & 6 & 5\end{pmatrix}\xrightarrow{x_{31}=9}\begin{pmatrix}6 & -1 & 1\\ 6 & -1 & 1\\ 6 & 6 & 6\end{pmatrix}\xrightarrow{x_{32}=1}\begin{pmatrix}0 & -1 & 1\\ 0 & -1 & 1\\ 0 & 0 & 0\end{pmatrix}\rightarrow$$

$$x_{33} = -1$$

$$\to \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \to \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Перейдемо до розв'язку цього прикладу методом (в). Для цього із загальної системи виділимо чотири підсистеми з відповідними множниками: підсистему першого роду

$$\left.\begin{aligned} 5S_2 + 5S_3 &\equiv 5 \\ 5S_1 + 5S_3 &\equiv 0 \\ 3S_1 + 3S_2 + 9S_3 &\equiv 12 \end{aligned}\right\} (\text{mod}15)$$

і три підсистеми другого роду:

$$\left.\begin{aligned} 5S_1 + 5x_{21} + 5x_{31} &\equiv 5 \\ 5x_{11} + 5S_2 + 5x_{31} &\equiv 0 \\ 3x_{11} + 3x_{21} + 3S_3 &\equiv 0 \end{aligned}\right\} (\text{mod}15),$$

$$\left.\begin{aligned} 5S_1 + 5x_{22} + 5x_{32} &\equiv 0 \\ 5x_{12} + 5S_2 + 5x_{32} &\equiv 0 \\ 3x_{12} + 3x_{22} + 3S_3 &\equiv 0 \end{aligned}\right\} (\text{mod}15),$$

$$\left.\begin{aligned} 5S_1 + 5x_{23} + 5x_{33} &\equiv 0 \\ 5x_{13} + 5S_2 + 5x_{33} &\equiv 0 \\ 3x_{13} + 3x_{23} + 3S_3 &\equiv 12 \end{aligned}\right\} (\text{mod}15).$$

Відповідно до алгоритму, викладеному вище, знаходимо значення S_i. Віднімаючи в першій підсистемі з першого рівняння друге, знаходимо $S_2 = S_1 + \frac{5}{5}$. Залежно від значення $\frac{5}{5}(\text{mod}15)$, що дорівнює 1, або –2, отримаємо два значення S_2. З другого рівняння знаходимо S_3, яке дорівнює або $-S_1$, або $2S_1$. З огляду на це, отже, можна розрізнити чотири випадки.

Випадок 1. $S_2 = S_1 + 1$, $S_3 = 2S_1$. Підставляючи ці значення в останнє рівняння першої системи, отримаємо значення $S_1 = 1$, $S_2 = S_3 = 2$, а потім, підставляючи їх у наступні підсистеми, знайдемо матрицю

$$X=\begin{pmatrix}2 & 1 & -2\\ 1 & 2 & -1\\ 2 & 0 & 0\end{pmatrix}.$$

Перевіримо це рішення.

$$B=\begin{pmatrix}-1 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & -4\end{pmatrix}\xrightarrow{x_{11}=2}\begin{pmatrix}1 & 2 & 2\\ 2 & 0 & 0\\ 2 & 0 & -4\end{pmatrix}\xrightarrow{x_{12}=1}\begin{pmatrix}2 & 3 & 3\\ 2 & 1 & 0\\ 2 & 1 & -4\end{pmatrix}\xrightarrow{x_{13}=-2}\begin{pmatrix}0 & 1 & 1\\ 2 & 1 & -2\\ 2 & 1 & -6\end{pmatrix}\xrightarrow{x_{21}=1}$$

$$\xrightarrow{x_{22}=2}\begin{pmatrix}1 & 1 & 1\\ 0 & 2 & -1\\ 3 & 1 & -6\end{pmatrix}\xrightarrow{x_{23}=-1}\begin{pmatrix}1 & 0 & 1\\ 2 & 1 & 1\\ 3 & 3 & -6\end{pmatrix}\xrightarrow{x_{31}=2}\begin{pmatrix}1 & 0 & 0\\ 1 & 0 & 0\\ 3 & 3 & -7\end{pmatrix}\to\begin{pmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{pmatrix}.$$

Випадок 2. $S_2 = S_1 - 2$, $S_3 = 2S_1$. Підставляючи ці значення в третє рівняння першої підсистеми, отримаємо таке співвідношення

$$3S_1 + 3S_1 - 6 + 18S_1 \equiv 12 \pmod{15},$$

звідки знаходимо $S_1=2$, $S_2=0$, $S_3=4$. Підставляючи ці значення в другу підсистему, отримаємо значення першого стовпця матриці X, а саме: $x_{11}=1, x_{21}=0, x_{31}=-1$. Із третьої підсистеми отримаємо значення другого стовпця матриці X, а саме: $x_{12}=-1, x_{22}=-3, x_{32}=1$. Під час підрахунку третього стовпця матриці виникають деякі проблеми.

За визначенням, $x_{13}=S_1-x_{11}-x_{12}\equiv 2 \pmod 3$. Визначимо x_{23}. З одного боку, $x_{23}=S_2-0-(-3)\equiv 0 \pmod 3$. З іншого — з третього рівняння підсистеми 4 випливає, що $3x_{23}=12-3S_3-3x_{13}$. Загальним рішенням буде $x_{23}=3$. А тепер визначимо x_{33}. За визначенням, $x_{33}=S_3-1+1\equiv 4 \pmod 5$. А з першого рівняння підсистеми 4 необхідно, щоб $5x_{33}=-5S_1-5x_{23}$, тобто $x_{33}=1$, або 4. Загальним рішенням є $x_{33}=4$.

У результаті отримаємо матрицю $X = \begin{pmatrix} 1 & -1 & 2 \\ 0 & -3 & 3 \\ -1 & 1 & 4 \end{pmatrix}$.

Перевіримо цей розв'язок.

$$B = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -4 \end{pmatrix} \xrightarrow{x_{11}=1} \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & -4 \end{pmatrix} \xrightarrow{x_{12}=-1} \begin{pmatrix} -1 & 0 & 0 \\ 1 & -1 & 0 \\ 1 & -1 & -4 \end{pmatrix} \xrightarrow{x_{13}=2} \begin{pmatrix} 1 & 2 & 2 \\ 1 & -1 & 2 \\ 1 & -1 & -2 \end{pmatrix} \xrightarrow{x_{22}=-3}$$

$$\xrightarrow{x_{23}=3} \begin{pmatrix} 1 & -1 & 2 \\ -2 & -4 & -1 \\ 1 & -4 & -2 \end{pmatrix} \xrightarrow{x_{31}=-1} \begin{pmatrix} 1 & -1 & 5 \\ 1 & -1 & 2 \\ 1 & -4 & 1 \end{pmatrix} \xrightarrow{x_{32}=1} \begin{pmatrix} 0 & -1 & 5 \\ 0 & -1 & 2 \\ 0 & -5 & 0 \end{pmatrix} \xrightarrow{x_{33}=4} \begin{pmatrix} 0 & 0 & 5 \\ 0 & 0 & 2 \\ 1 & -4 & 1 \end{pmatrix} \rightarrow \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Випадок 3. $S_2 = S_1 + 1$, $S_3 = -S_1$. Підставляючи ці значення в останнє рівняння першої системи, отримаємо значення $S_1 = -3$, $S_2 = -2, S_3 = 3$. Підставивши їх у другу й третю підсистеми, отримаємо значення першого та другого стовпців матриці X, а саме:

$$x_{11} = 0,\ x_{21} = 2,\ x_{31} = 2\ ,\ x_{12} = -2,\ x_{22} = -1,\ x_{32} = 4\ .$$

Значення третього стовпця матриці X отримаємо іншим способом. За визначенням $x_{13} = -1$, $x_{23} = S_2 - x_{21} - x_{22} = -3$. З іншого боку, із четвертої підсистеми знаходимо, що $3x_{23} = 12 - 3S_3 - 3x_{13} = 6$, тобто $x_{23} = 2$, або -3. Загальним є значення $x_{23} = -3$. З першого рівняння цієї системи знаходимо, що $5x_{33} = 5S_1 - 5x_{23} = 30 \equiv 0 \,(\mathrm{mod}15)$. Отже, $x_{33} = -3$, або 0, або 3. А за визначенням $x_{33} = S_3 - x_{31} - x_{32} = -3$, тобто загальним є $x_{33} = -3$.

Остаточним розв'язком буде матриця $X = \begin{pmatrix} 0 & -2 & -1 \\ 2 & -1 & -3 \\ 2 & 4 & -3 \end{pmatrix}$.

Неважко перевірити, що це є розв'язком задачі.

Випадок 4. $S_2 = S_1 - 2$, $S_3 = -S_1$. Підставляючи ці значення в останнє рівняння першої системи, отримаємо значення $S_1 = -1$, $S_2 = -3, S_3 = +1$. Міркуючи так само, як й у випадку 3, отримаємо значення матриці $X = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & -2 \\ 0 & 1 & 0 \end{pmatrix}$. Легко переконатися, що це є розв'язком.

Для довільних значень матриці B не завжди існує розв'язок. У цьому переконаємося на такому прикладі.

Приклад 2. Нехай $m = n = 5$, $k_1 = 2$, $k_2 = 3$, $p = 2$, а матриця B має вигляд

$$B = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -2 & 0 & 0 \\ 0 & 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 0 & -1 \end{pmatrix}.$$

Утворимо відповідні підсистеми для цієї матриці:

систему першого роду

$$\left.\begin{aligned} a_1 &= 25S_1 + 5S_2 + 5S_3 + 5S_4 + 5S_5 \equiv 5 \\ a_2 &= 5S_1 + 25S_2 + 5S_3 + 5S_4 + 5S_5 \equiv 5 \\ a_3 &= 3S_1 + 3S_2 + \ \ldots\ + 3S_4 + 3S_5 \equiv 6 \\ a_4 &= 3S_1 + 3S_2 + 3S_3 + \ \ldots\ + 3S_5 \equiv 6 \\ a_5 &= 3S_1 + 3S_2 + 3S_3 + 3S_4 \quad \ldots \quad \equiv 3 \end{aligned}\right\} (\mathrm{mod}15)$$

і п'ять підсистем другого роду теж за модулем 15

$$\left.\begin{aligned} 5S_1 + 5x_{21} + 5x_{31} + 5x_{41} + 5x_{51} &= 5 \\ 5x_{11} + 5S_2 + 5x_{31} + 5x_{41} + 5x_{51} &= 0 \\ 3x_{11} + 3x_{21} + 3S_3 + 3x_{41} + 3x_{51} &= 0 \\ 3x_{11} + 3x_{21} + 3x_{31} + 3S_4 + 3x_{51} &= 0 \\ 3x_{11} + 3x_{21} + 3x_{31} + 3x_{41} + 3S_5 &= 0 \end{aligned}\right\},$$

$$\left.\begin{array}{l} 5S_1+5x_{22}+5x_{321}+5x_{42}+5x_{52}=0\\ 5x_{12}+5S_2+5x_{32}+5x_{42}+5x_{52}=5\\ 3x_{12}+3x_{22}+3S_3+3x_{42}+3x_{52}=0\\ 3x_{12}+3x_{22}+3x_{32}+3S_4+3x_{52}=0\\ 3x_{12}+3x_{22}+3x_{32}+3x_{42}+3S_5=0 \end{array}\right\},$$

$$\left.\begin{array}{l} 5S_1+5x_{23}+5x_{33}+5x_{43}+5x_{53}=0\\ 5x_{13}+5S_2+5x_{33}+5x_{43}+5x_{53}=0\\ 3x_{13}+3x_{23}+3S_3+3x_{43}+3x_{53}=6\\ 3x_{13}+3x_{23}+3x_{33}+3S_4+3x_{53}=0\\ 3x_{13}+3x_{23}+3x_{33}+3x_{43}+3S_5=0 \end{array}\right\},$$

$$\left.\begin{array}{l} 5S_1+5x_{24}+5x_{34}+5x_{44}+5x_{54}=0\\ 5x_{14}+5S_2+5x_{34}+5x_{44}+5x_{54}=0\\ 3x_{14}+3x_{24}+3S_3+3x_{44}+3x_{54}=0\\ 3x_{14}+3x_{24}+3x_{34}+3S_4+3x_{54}=6\\ 3x_{14}+3x_{24}+3x_{34}+3x_{44}+3S_5=0 \end{array}\right\},$$

$$\left.\begin{array}{l} 5S_1+5x_{25}+5x_{35}+5x_{45}+5x_{55}=0\\ 5x_{15}+5S_2+5x_{35}+5x_{45}+5x_{55}=0\\ 3x_{15}+3x_{25}+3S_3+3x_{45}+3x_{55}=0\\ 3x_{15}+3x_{25}+3x_{35}+3S_4+3x_{55}=0\\ 3x_{15}+3x_{25}+3x_{35}+3x_{45}+3S_5=3 \end{array}\right\}.$$

Зробивши кроки 1 і 2 етапу 1 знаходимо $S_1=S_2$, $S_4=S_3$, $S_5=S_3+1$. Крок 3 зробити неможливо, тому що при підстановці значень $S_4=S_3+\frac{a_3-a_4}{3}$ і $S_5=S_3+\frac{a_3-a_5}{3}$ у перше рівняння отримаємо тотожність

$$25S_1+5S_1+5S_3+5S_3+5S_3+5\left(\frac{a_3-a_4}{3}+\frac{a_3-a_5}{3}\right)\equiv a_1(\mathrm{mod}15).$$

Це можливо тільки за однієї умови $2a_3-a_4-a_5=\frac{3}{5}a_1$. Якщо ця умова не виконується, задача не має розв'язку. У цьому прикладі ця умова виконується, і

задача має кілька розв'язків, тому що з рівняння a_5 випливає, що S_1 й S_3 можуть приймати різні значення за умови $S_1 + S_3 = 3(\text{mod}5)$.

Якщо задати, наприклад, значення S_1=1, S_3=2, то отримаємо інші значення $S_2 = 1, S_4 = 2, S_5 = 3$. Підставляючи ці значення в $j-$і ($j = 1,2,3,4$) підсистеми другого роду та дотримуючись схеми етапу 2, отримаємо значення перших 4-х стовпців матриці *X*: (1,2,2,2,3), (2,1,2,2,3), (0,0,0,2,3), (0,0,2,0,3). А 5-й стовпець отримаємо таким чином.

За визначенням

$$x_{15} = x_{25} = S_1 - 1 - 2 = -2 \ (\text{mod}3),$$

$$x_{35} = x_{45} = S_3 - 2 - 2 - 2 = 1 \ (\text{mod}5),$$

$$x_{55} = S_3 - 3 - 3 - 3 = 1 \ (\text{mod}\ 5).$$

Із цих значень x_{35}, x_{45}, x_{55} задовольняють останнім трьом рівнянням підсистеми 5. З іншого боку, з першого рівняння цієї підсистеми випливає $x_{55} = -1(\text{mod}\ 3)$. Загальним розв'язком є $x_{55} = 11$.

Таким чином, матриця розв'язку набуде вигляду

$$X = \begin{pmatrix} 1 & 2 & 0 & 0 & -2 \\ 2 & 1 & 0 & 0 & -2 \\ 2 & 2 & 0 & 2 & 1 \\ 2 & 2 & 2 & 0 & 1 \\ 3 & 2 & 2 & 3 & 11 \end{pmatrix}.$$

Випадки виродження першого рівняння у тотожність виникають за таких параметрів: $n + p - 1 \equiv 0(\text{mod}\, k_1)$, $a_1 = a_2 = ... = a_p$, $m - p = k_2$. У такій ситуації S_{p+1} залежить від S_1, і розв'язок задачі не однозначний. Крім цього, на a_i накладають певні обмеження на кшталт вищезгаданих, без яких задача не має розв'язку.

Подібні задачі для сейфів більш, ніж з двома типами замків розв'язуються аналогічно за алгоритмом, описаним на етапах 1, 2.

4.6. Висновки

Розглянуті задачі про математичний сейф можна віднести до задач, які носять назву «позиційні ігри». У них необхідно за допомогою послідовності ходів, якими є кількість поворотів ключа у відповідних замках сейфа, досягти положення, коли всі замки досягнуть певного стану, за якого сейф вважається відкритим. Такі задачі зводяться до розв'язання системи лінійних рівнянь у класах лишків за певним модулем. Наведені умови, які дають відповідь про існування чи неіснування розв'язку задачі. Результати можуть застосовуватися у теорії захисту інформації.

Список літератури

1. Kryvyi S. L. Algorithms for solution of systems of linear diophantine equations in residue fields. Cybernetics and Systems Analysis. 2007. 43 (2). P. 171−178.
2. Агаи Аг Гамиш Якуб, Донец Г. А. Задача о математическом сейфе на матрицах. Теорія оптимальних рішень. 2013. С. 124–130.
3. Гурин А. Л. Методы решения задач о математических сейфах на матрицах с разными типами замков. Кибернетика и системный анализ. 2019. № 2.

5. ЕКСТРЕМАЛЬНІ ЗАДАЧІ НА КОМБІНАТОРНИХ КОНФІГУРАЦІЯХ

Г. П. Донець, Е. І. Ненахов

Анотація. У роботі досліджуються екстремальні задачі на комбінаторних конфігураціях. Вивчаються деякі структурні властивості допустимих областей таких задач, а також сформульовано ряд тверджень, які дають можливість побудувати методи розв'язання комбінаторної задачі із застосуванням теорії графів. Описано алгоритми знаходження глобального екстремуму в оптимізаційних задачах з лінійною, квадратичною та дробово-лінійною цільовими функціями на комбінаторних конфігураціях.

Annotation. In this paper, extremal problems on combinatorial configurations are investigated. Some structural properties of feasible regions of such problems are studied, and also a number of statements are formulated which enable to construct methods for solving combinatorial problem using graph theory. The algorithms for finding global extremum in optimization problems with linear, quadratic and fractional-linear objective functions on combinatorial configurations are described.

5.1. Побудова графа комбінаторної конфігурації

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

5.1.1. Використання графів для розв'язування екстремальних комбінаторних задач

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

цікавих властивостей; при їх вивченні виникають задачі, що представляють інтерес не тільки для теорії графів, комбінаторики, топології і геометрії, а і для теорії лінійного програмування.

Використання властивостей графів комбінаторних конфігурацій можуть послужити підвищенню ефективності «традиційних» і розробці нових методів розв'язування екстремальних задач комбінаторної оптимізації. Комбінаторні моделі можуть бути застосовані для представлення оптимізаційних задач, що виникають при оптимальному розміщенні на графах. Теорія графів вивчає екстремальні властивості графів, розглядаючи множину його граней усіх розмірностей як деякий комплекс. Але при розв'язанні таких задач виникають проблеми, пов'язані зі складністю математичних моделей, великим обсягом інформації і т. д., оскільки більшість задач на комбінаторних конфігураціях є *NP*-повними. Більшість задач на графах стосується визначення компонентів зв'язності, пошуку маршрутів, відстаней тощо. Проте при розв'язанні прикладних задач відповідні їм графи, що залучаються, мають достатньо великий обсяг, а аналіз можливий лише із застосуванням сучасної обчислювальної техніки. У цій роботі досліджується екстремальна задача на комбінаторних конфігураціях. На підставі встановленого взаємозв'язку між задачами на комбінаторних множинах і графами многогранників комбінаторних множин вивчаються деякі структурні властивості допустимої області, а також сформульовано ряд тверджень, які дають можливість побудувати методи розв'язання комбінаторної задачі з використанням графів. Зокрема в роботі [1], описаний спосіб побудови гамільтонового шляху усередині гіперграней. У статті [2] ставиться задача побудови гамільтонового шляху між гіперграними, тобто, як побудувати гамільтонів шлях, додавши до двох графів без спільних вершин третій граф, що не має спільних вершин з першими двома, якщо гамільтонів шлях побудовано для перших двох графів.

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

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

Визначення 5.1. Граф, усі ребра якого неорієнтовані, називають неорієнтованим графом, а граф, усі ребра якого орієнтовані, — орієнтованим графом, або орграфом.

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

Шляхом в орієнтованому графі називається односторонній орієнтований маршрут. Очевидно, що маршрут і для звичайних графів однозначно визначається послідовністю вершин: $v_1 v_{i_1} v_{i_2} \ldots v_{i_{n-1}} v_2$.

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

Визначення 5.2. Гамільтоновим шляхом у графі називають простий шлях, який проходить усі вершини графа.

5.1.2. Генерування розміщень

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

генерування розміщень другого типу для послідовності довжини 3 з елементами з множини $X = \{1, 2, 3, 4\}$. На графі можна представити ці об'єкти у вигляді рис. 5.1.

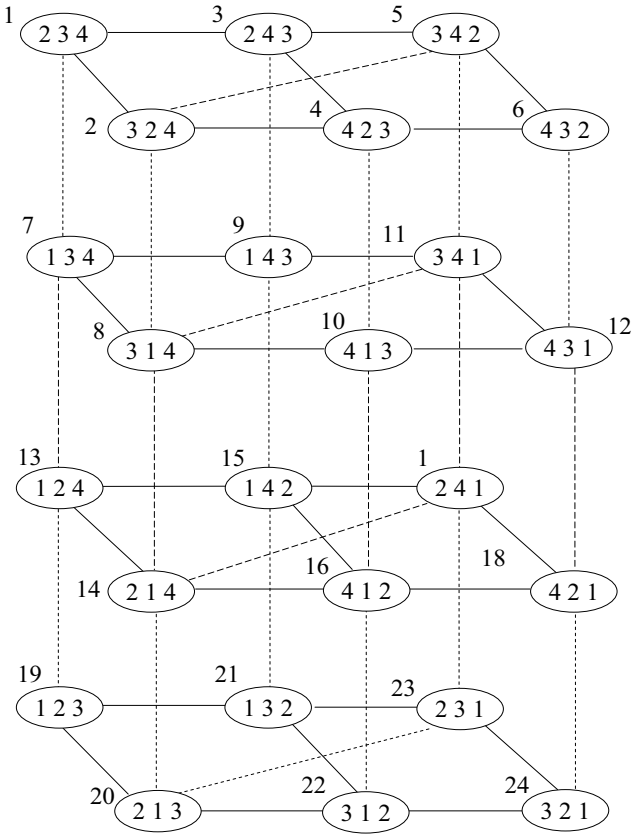


Рис. 5.1. Граф множини розміщень для $n = 4$ та $m = 3$

Цю побудову можна представити також і рекурентним методом. Отже, маємо два числа $n > m \geq 1$ і треба побудувати граф розміщень з $A_n(m) = [n]_m$ вершин. На початку розглянемо випадок $n = m + 1$ (на рис. 5.1). Будуємо граф

перестановок для $G_m(P)$, для чого можемо скористатися одним із представлених вище методів. На рис. 5.1 це $G_3(P)$, (вершини 19–24). Далі робимо $n-1$ копій цього графа. У першій копії (вершини 13–18) робимо скрізь заміну $n-1$ на n (3 на 4) та з'єднуємо ребрами (пунктирними) відповідні вершини (13, 19), (14, 20) і т. д. Далі за індукцією в *i*-й копії ($n-1 \ge i \ge 1$) всюди робимо заміну елементів $n-i$ на $n-i+1$, потім з'єднуємо ребрами відповідні вершини *i*-ї та $(i-1)$-ї копій. Цим і завершується побудова графа розміщень $G_m(A_{m+1})$. Як бачимо, суміжність вершин в цьому графі має двоякий вигляд — в $G_3(P)$ та його копіях, як і в перестановках, завдяки одній транспозиції, а вершин з різних копій — відмінністю тих двох елементів, які замінюються на одному місці. Оскільки для $n=m+1$ число розміщень дорівнює $m(m-1)(m-2)2=m!$, то очевидно, що при цьому граф розміщень співпадає з графом перестановок. Загальна схема рекурентної побудови графа розміщень для довільних *n* та *m* дещо відрізняється.

Почнемо з побудови графа $G_1(A_n)$. Це ізольовані вершини з номерами $1, 2, \ldots, n-1, n$. Побудова графа $G_m(A_n)$ ведеться за такою індукцією: спочатку будуємо граф $G_1(A_{n-m+1})$. Для цього припускається, що побудовано граф розміщень $G_i(A_{n-m+i})$ $(m \ge i \ge 1)$, де довжина коду розміщень дорівнює *i*. Далі робимо $i+1$ копію графа $G_i(A_{n-m+i})$. У j-й копії ($i+1 \ge j \ge 1$) справа в кожному розміщенні дописуємо число *j*. При цьому, якщо таке число в розміщені є зліва, то воно замінюється числом $i+1$. З'єднуємо ребрами відповідні вершини j-ї та $(j-1)$-ї копій ($j>1$). Якщо такі дії виконані для кожного j від 1 до $i+1$, то тим самим побудовано граф $G_{i+1}(A_{n-m+i+1})$.

Коли довжина коду розміщень стане рівною *m*, то це означатиме, що отримано граф розміщень $G_m(A_n)$. Зазначимо, що відомий граф $G_1(A_3)=1, 2, 3$. Нижче наведено таблицю, яка ілюструє дію цього алгоритму для побудови $G_3(A_5)$.

*Табл. 5.1.*Побудова підграфів

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

5.1.3. Генерування розбиття множини та числа

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

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

$$
\begin{aligned}
&B_1 \cup \{n\}, B_2, \ldots, B_k \\
&B_1 B_2, \cup \{n\}, \ldots, B_k \\
&\ldots\ldots\ldots\ldots\ldots\ldots \\
&B_1 B_2, \ldots, B_k \cup \{n\} \\
&B_1 B_2, \ldots, B_k \{n\}
\end{aligned}
\qquad (5.1)
$$

Тоді рекурентний метод генерування розбиття множини полягає в такому: якщо отримано список L_{n-1} усіх розбиттів множини $\{1, \ldots, n-1\}$, то список L_n усіх розбиттів множини $\{1, \ldots, n\}$ одержуватимемо, замінюючи кожне розбиття σ у списку L_{n-1} на відповідну йому послідовність (5.1). Зазначимо, що при цьому можна гарантувати, що розбиття, які йдуть одне за одним, мало відрізнятиметься одне від одного, точніше кажучи, що кожне наступне розбиття у списку утворюється з попереднього за допомогою видалення деякого елементу з певного блоку і додавання його в інший блок або створення з нього одноелементного блоку.

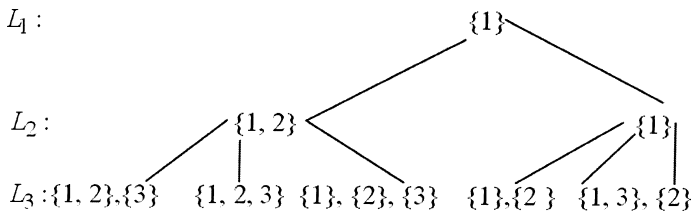


Рис. 5.2. Побудова списків L_1, L_2, L_3

Дійсно, послідовні розбиття послідовності (5.1) відповідають цій умові. Якщо порядок послідовності зробити зворотним $\{3, 3\}$ для кожного другого розбиття списку L_{n-1}, то елемент n переміщуватиметься почергово вперед і назад, і розбиття «на стику» послідовностей, утворених із сусідніх розбиттів списку L_{n-1}, мало відрізнятимуться одне від одного (за умови, що сусідні розбиття списку L_{n-1} мало відрізняються одне від одного). На рис. 5.2 показано

побудову списку L_n для множини елементів 1, 2, 3. Цей спосіб генерування будує спочатку розбиття $\{1,\ldots,n\}$, потім здійснюється переміщення «активного» елементу j у сусідній блок — попередній або подальший (у останньому випадку може виникнути необхідність створення нового блоку елементів), а потім визначення активного елементу в новоутвореному розбитті. З описаної побудови випливає, що цей елемент переміщається тільки тоді, коли всі елементи, що більші за нього, досягають свого крайнього лівого або правого положення; точніше, активний елемент j^* є таким найменшим елементом.

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

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

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

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

де i — вертикальна вісь, j — горизонтальна вісь.

Точковий граф являє собою наступну графічну структуру (рис. 5.3), де частини розбиття розміщуються, як правило, зверху вниз за спаданням.

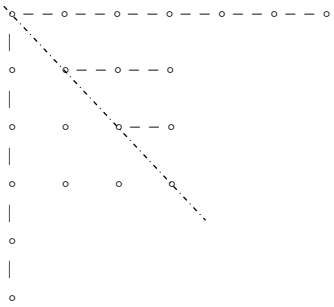


Рис. 5.3. Графічна структура точкового графа

Таким чином, методи генерування комбінаторних конфігурацій дають можливість представити їх у вигляді графа комбінаторних конфігурацій, який відображає елементи конфігурації і по якому можна рухатися залежно від значення функції. Цей граф буде застосовано для побудови методів розв'язання екстремальних комбінаторних задач.

5.2. Оптимізація лінійної та квадратичної функцій на перестановках

5.2.1. Загальна постановка задачі на комбінаторних конфігураціях

Загальна задача комбінаторної оптимізації полягає у відшуканні екстремуму довільної цільової функції на довільних комбінаторних конфігураціях. Як правило, при розв’язанні класу таких задач досліджується можливість їх лінеаризації, тобто побудови опуклої оболонки допустимих розв’язків задачі. Перехід від параметричної форми задання опуклого многогранника до аналітичної має велике значення для задач дискретної оптимізації, оскільки дозволяє сформулювати їх у термінах лінійного програмування. Але застосування методів лінійного програмування не завжди відповідає характеру екстремальних задач. Застосування теорії графів до розв’язання таких задач дає в багатьох випадках можливість отримати розв’язок з меншими витратами комп’ютерних ресурсів. Підзадачею вище-

сформульованої задачі може бути визначення гамільтонового шляху, який визначає зміну значення цільової функції на множині комбінаторних конфігурацій. Многогранник гамільтонових циклів графа є гранню многогранника гамільтонових циклів повного графа. Отже, розглянемо побудову послідовності значень лінійної цільової функції на графі перестановок. При цьому не обов'язково будувати повний граф, достатньо знайти таку часткову упорядкованість, яка дозволить виділити такі спадаючі (або зростаючі) за значеннями функції шляхи, які приведуть нас до екстремуму функції. Це зобов'язує детальніше вивчати цільову функцію та враховувати властивості і структуру послідовності перестановок, на якій розглядається задача.

Розглянемо екстремальну задачу комбінаторної оптимізації вигляду:

$$Z\left(\Phi, P\left(A\right)\right) : max\left\{\Phi\left(a\right) \mid a \in P\left(A\right)\right\},$$

яка полягає в максимізації функції $\Phi\left(a\right)$ на множині перестановок $P\left(A\right)$, де $\Phi\left(a\right) = \sum_{j=1}^{n} c_j x_j$. Відомо, що можна сформулювати екстремальну задачу $Z\left(F, X\right)$ максимізації критерію $F\left(x\right)$ на множині X, причому кожній точці $a \in P_{nk}\left(A\right)$ відповідатиме точка $x \in X$, така, що $F\left(x\right) = \Phi\left(a\right)$.

$$Z\left(F, X\right) : max\left\{F\left(x\right) \mid x \in X\right\}, \tag{5.2}$$

де $F\left(x\right) = \sum_{j=1}^{n} c_j x_j$, X — не порожня множина в R^n, яка визначається таким чином: $X = vert\,\Pi\left(A\right)$, $\Pi = conv\,P\left(A\right)$.

Слід зазначити, що іноді є доцільним розв'язувати задачу вигляду:

$$x^* = \underset{x \in \Pi(A)}{arg\,max}\, F\left(x\right),$$

для значення функції $y^* = F\left(x^*\right)$. Так само має сенс розглядати різновид вищезазначеної задачі, де значення цільової функції знаходиться в інтервалі

$$F\left(\overline{x}\right) \le F\left(x\right) \le F\left(\overline{\overline{x}}\right).$$

Тоді задача набуде вигляду: визначити

$$\overline{x} = \underset{x \in \Pi(A)}{arg\,max}\, F(x) \text{ при } \overline{y} = F(\overline{x}),\ \overline{\overline{x}} = \underset{x \in \Pi(A)}{arg\,max}\, F(x) \text{ при } \overline{\overline{y}} = F(\overline{\overline{x}}) \qquad (5.3)$$

за умови $|\overline{x} - \overline{\overline{x}}| \to min$.

Розглядаємо перестановку як упорядковану вибірку елементів $a = (a_{i_1}, a_{i_2}, \ldots, a_{i_n})$, де $a_{i_j} \in A \quad \forall i_j \in N_n$, $\forall j \in N_n$, $i_s \neq i_t$, якщо $s \neq t \quad \forall s \in N_n$, $\forall t \in N_n$ з деякої мультимножини $A = \{a_1, a_2, \ldots, a_q\}$, яке характеризується основою $S(A) = \{e_1, e_2, \ldots, e_k\}$, де $e_j \in R^1, \forall j \in N_k$ і кратністю елементів $k(e_j) = r_j$, $j \in N_k$, $r_1 + r_2 + \ldots + r_k = q$ згідно з [3].

Множина перестановок з повтореннями з n дійсних чисел, серед яких k різних, називається загальною множиною перестановок і позначається $P_{nk}(A)$. Це множина впорядкованих n-вибірок з мультимножини A за умови $n = q > k$. При $n = k = q$ маємо множину перестановок без повторень. Позначимо її P_n. Очевидно, що $P_n(A) = P_{nn}(A)$. У тих випадках, коли не вказується вигляд множині перестановок, записуватимемо ці множини, як $P(A)$. Відомо [3], що опуклою оболонкою множини перестановок є переставний многогранник $\Pi(A) = conv\, P(A)$, множина вершин якого рівна множині $P(A)$ перестановок: $vert\, \Pi(A) = P(A)$.

Не втрачаючи загальності, упорядкуємо елементи мультимножини A у неспаданні:

$$a_1 \le a_2 \le \ldots \le a_n, \qquad (5.4)$$

і елементи його основи — у зростанні: $e_1 < e_2 < \ldots < e_k$. Тоді опуклою оболонкою загальної множини перестановок $P(A)$ є переставний многогранник $\Pi(A) = conv\, P(A)$, який описується відомою системою лінійних нерівностей:

$$\begin{cases} \sum_{j=1}^{n} x_j \le \sum_{j=1}^{n} a_j, \\ \sum_{j=1}^{i} x_{\alpha_j} \ge \sum_{j=1}^{i} a_j, \end{cases}$$

де $\alpha_j \in N_n, \alpha_j \neq \alpha_t, \forall j \neq t, \forall j, t \in N_i, \forall i \in N_n$, а $P(A) = vert\, \Pi(A)$.

5.2.2. Оптимізація лінійної функції на перестановках

Розглянемо підхід до розв'язання задач, що ґрунтується на впорядковуванні значень цільової лінійної функції $F(x)$ і побудові гамільтонового шляху для точок, у яких ці значення досягаються, а потім у застосуванні методу дихотомії до визначеного гамільтонового шляху. Далі під задачею $Z(\Phi, P(A))$ розуміємо задачу $Z(F, X)$.

Для побудови методу перш за все на початковому етапі необхідно визначити початкову точку. Розглянемо факт у вигляді твердження.

Твердження. Якщо для елементів мультимножини A і коефіцієнтів цільової функції задачі

$$extr\left\{ f(x) = \sum_{j=1}^{n} c_j x_j \mid x \in vert\, \Pi(A) \right\}$$

виконуються відповідно до умови (5.4) і $c_{i_1} \le c_{i_2} \le \ldots \le c_{i_n}$, $i_n \in N_n$, то максимум функції $f(x)$ на допустимій множині досягається у точці $x^* = \left(x^*_{i_1}, \ldots, x^*_{i_n}\right) \in vert\, \Pi(A)$, яка задається таким чином:

$$x^*_{i_j} = a_j \;\forall\; j \in N_n,$$

а мінімум відповідно в точці $y = (y_1, y_2, \ldots, y_n)$, де $y_{i_{j+1}} = a_{n-j} \;\forall\; j \in N_{n-1} \bigcup \{0\}$.

Послідовності перестановок $P(A)$ визначаємо як вершини графа $G(P_n)$, що відповідають усім точкам множини перестановок. Для такого графа дві вершини визнаватимемо суміжними, якщо коди відповідних перестановок відрізняються одноразовою транспозицією двох елементів. З'єднуючи відповідні вершини по ходу генерації перестановок, одержуємо спочатку неорієнтований граф. Немає необхідності (як можна впевнитися далі) з'єднувати всі суміжні вершини, а тільки ті, що генеруються у цей момент і будь-який гамільтонів ланцюг відповідає деякому варіанту генерування усіх перестановок. Якщо в кожній вершині графа знайти відповідне значення заданої цільової функції $F(x)$, то відносно значень функції можна побудувати

орієнтований граф, зорієнтувавши його ребра в напрямі від більшого значення функції у суміжних вершинах до меншого. Досить актуальною для розробки нових методів розв'язування задачі є питання побудови гамільтонового шляху, уздовж якого всі значення функції строго впорядковані за спаданням (збільшенням). Очевидно, що кінцеві точки гамільтонового шляху визначають екстремальні точки (перестановки). Тоді є важливим наступний факт, який можна сформулювати у вигляді леми.

Лема 5.1. Якщо з перестановки $p_1 = (i_1, i_2, \ldots, i_n) \in P(A)$ одержана перестановка $p_2 \in P(A)$ транспозицією двох чисел $i_k < i_l$, де $k < l$, то $F(p_1) \geq F(p_2)$.

Розглянемо лінійну функцію $F(x,c) = (c,x) = \sum_{i=1}^{n} c_i x_i$, де $c = (c_1, c_2, \ldots, c_n)$ множина довільних чисел, що визначають коефіцієнти цільової функції, а $x = (x_1, x_2, \ldots, x_n)$ — перестановка чисел $(1, 2, \ldots, n)$. Розглянемо u-перестановку із симетричної групи перестановок S_n. Нехай $F(x,c,u) = \sum_{i=1}^{n} c_i x_{u(i)}$. Основна проблема комбінаторної оптимізації полягає в тому, щоб знайти таку перестановку u_0, що $F(x,c,u_0) \leq F(x,c,u)$ для довільного $u \in H$, де H — довільна не порожня підмножина симетричної групи S_n. Якщо $t \in S_n$, то позначимо $c^t = \left(c_{t(1)}, c_{t(2)}, \ldots, c_{t(n)}\right)$, $x^t = \left(x_{t(1)}, x_{t(2)}, \ldots, x_{t(n)}\right)$. Тоді $F(x,c) = F(c^t, x^t) = (c,x) = (c^t, x^t)$. Аналогічно $F(x,c,u) = F(x^t, c^t, u')$, де $u' = t^{-1}ut$ для всякого u з S_n. Отже, при розв'язанні основної проблеми завжди можна замінити пару (x,c) на пару (x^t, c^t). Зокрема завжди можна вважати:

$$c_1 \leq c_2 \leq \ldots \leq c_n, \qquad \text{(а)}$$

або

$$c_1 \geq c_2 \geq \ldots \geq c_n. \qquad \text{(б)}$$

Якщо $H = S_n$, то проблема розв'язана [4]. Доведено, якщо є (а), то максимум функції $F(x,c,u)$ досягається для перестановки $u_0 = (1, 2, \ldots, n)$, а мінімум — для перестановки $u^* = (n, n-1, \ldots, 2, 1)$. У подальших дослідженнях передбачатиметься, що для коефіцієнтів $c_i \; (1 \le i \le n)$ завжди є (а). Розглянемо тепер детально структуру відповідного графа для невеликого значення n. Для $n = 3$ граф, зображений на рис. 5.4, де дуга, що виходить з перестановки p_i і заходить у перестановку p_j рівносильна співвідношенню $F(x, c, p_1) \ge F(x, c, p_2)$.

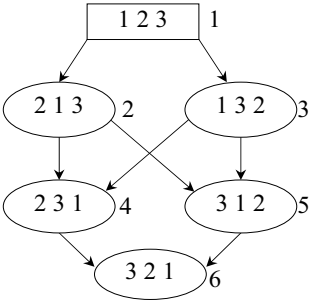


Рис. 5.4. Структура значень $F(x)$ для $n = 3$

Зазначимо, що на рис. 5.4 дуги відображають зміну значень цільової функції, а вершини графа побудовано відповідно до рекурсивного методу. Доведемо, що граф однозначно відображає впорядкування значень функцій. Для цього достатньо зробити безпосереднє обчислення перестановок.

Наприклад $-\dfrac{\begin{matrix}1 & 3 & 2\end{matrix}}{\begin{matrix}1 & 2 & 3\\ 0 & 1 & -1\end{matrix}}$. Різниця відповідає різниці значень функції $c_2 - c_3 \le 0$, що відповідає дійсності. На цьому графі немає з'єднань між перестановками $(1\ 3\ 2)$ і $(2\ 1\ 3)$, а так само між $(2\ 3\ 1)$ і $(3\ 1\ 2)$. Цим парам відповідає різниця $(1 - 2\ 1)$, що рівносильна $c_1 - 2c_2 + c_3$, значення якої не може бути визначено однозначно.

5.2.3. Оптимізація квадратичної функції на множині перестановок

Нехай дано задачу комбінаторної оптимізації: $Z(\Phi, P(A)) : \max\{\Phi(a) \mid a \in P(A)\}$, яка полягає в максимізації квадратичної функції $\Phi(a)$ виду:

$$\Phi(a) = \sum_{j=1}^{n} c_j x_j^2 + \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} x_i x_j \tag{5.5}$$

на множині перестановок $P(A)$. Тоді основна задача $Z(F, X)$ полягатиме в максимізації критерію $F(x)$ на множині X, причому кожній точці $a \in P_{nk}(A)$ відповідатиме точка $x \in X$, така, що виконується рівність $F(x) = \Phi(a)$:

$$Z(F, X) : \max\{F(x) \mid x \in X\}\ , \tag{5.6}$$

де $F(x) = \sum_{j=1}^{n} c_j x_j^2 + \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} x_i x_j$, X — непорожня множина в R^n, яка визначається таким чином:

$$X = vert\Pi(A)\ ,\ \Pi = convP(A)\ .$$

Отже, точка, що визначає максимум квадратичної функції буде однією з вершин переставного многогранника.

Тому послідовність перестановок $P(A)$ ототожнюватимемо з вершинам графа $G(P_n)$, що відповідають усім точкам множини перестановок. Тоді, врахувавши властивості графів, суміжність вершин визначається одноразовою траспозицією двох елементів вершини. З'єднуючи відповідні вершини, одержуємо граф, і будь-який гамільтонів ланцюг відповідає деякому варіанту генерування усіх перестановок. Якщо кожній вершині графа призначити деяке значення заданої цільової функції $F(x)$, то відносно значень функції можна побудувати відповідний граф, зорієнтувавши його.

Кількість транспозицій у графі буде рівна:

$$C_n^2 = \frac{n(n-1)}{2} \tag{5.7}$$

Алгоритм розв'язування оптимізаційної задачі з квадратичною функцією цілі полягає в наступному. Розглянемо транспозиції відповідних елементів, представляючи функцію у вигляді добутку:

$$x_i \leftrightarrow x_{i+1} \quad f_{ii+1} = (x_i - x_{i+1})\left(a_1x_1 + a_2x_2 + ... + a_ix_i + a_{i+1}x_{i+1}\right) \tag{5.8}$$

Умовно розділимо праву частину функції f_{ii+1} на дві й окремо розглянемо дві задачі:

$$f_{ii+1} \to \begin{cases} f^1_{ii+1} = (x_i - x_{i+1}), \\ f^2_{ii+1} = \left(a_1x_1 + a_2x_2 + ... + a_ix_i + a_{i+1}x_{i+1}\right) \end{cases}$$

У випадку функції декількох змінних можна розділити вплив аргументів на загальний результат, тобто

$$f(x_1, x_2, ..., x_n) = \sum_{i=1}^{n} f_i(x_i).$$

Задача 1. Знайти максимум функції:

$$\max \sum f^1_{ii+1}, \text{ де } f^1_{ii+1} = (x_i - x_{i+1}),\ i = \overline{1:n}$$

Розв'язок задачі рівний $(x_1, x_2, ..., x_i, x_{i+1})$, за умови $(x_1 > x_2 > ... > x_i > x_{i+1})$, який визначатимемо, як 1-й опорний розв'язок.

Задача 2. Знайти максимум функції:

$$\max \sum f^2_{ii+1}, \text{ де } f^2_{ii+1} = \left(a_1x_1 + a_2x_2 + ... + a_ix_i + a_{i+1}x_{i+1}\right),\ i = \overline{1:n}.$$

Для знаходження максимуму функції необхідно скористатися 1-м опорним розв'язком першої задачі $(x_1, x_2, ..., x_i, x_{i+1})$. Послідовно підставляємо перший опорний розв'язок $(x_1, ..., x_i, x_{i+1})$ в $f^2_{ii+1} = \left(a_1x_1 + a_2x_2 + ... + a_ix_i + a_{i+1}x_{i+1}\right),\ i = \overline{1:n}$.

Якщо $f^2_{ii+1} > 0$, то функція зростає, відповідно перший опорний розв'язок $(x_1, x_2, ..., x_i, x_{i+1})$ буде оптимальним.

Якщо $f^2_{ii+1} < 0$ при розв'язку $(x_1, x_2, ...x_k, ...x_m, ..., x_i, x_{i+1})$, то знаходимо такий x_k, де $x_k < x_m$, при якому функція f^2_{ii+1} зростає. Для цього здійснюємо перестановку x_k на x_m, відповідно наступний опорний розв'язок буде $(x_1, x_2, ...x_m, ...x_k, ..., x_i, x_{i+1})$. Перестановки здійснюються доти, поки не буде зроблена перевірка всіх транспозицій. На кожному кроці значення змінних або

залишатимуться без змін, або мінятимуться місцями для забезпечення максимізації функції.

Приклад. Дано функцію

$$F(x)=\frac{1}{2}(2x_1-x_2-x_3+x_4)^2+(x_2+x_3)^2-\frac{1}{2}(x_3-x_2+2x_4)^2+2x_4^2$$

на множині перестановок $P(A)$, де $A=(1,2,3,4)$. Необхідно знайти максимальне значення квадратичної функції.

Відповідно до вказаної вище формули (5.7), знаходимо кількість транспозицій: $C_4^2=6$. Розглянемо транспозиції відповідних елементів, представляючи функцію у вигляді добутку (5.8):

1. $x_1\leftrightarrow x_2$ $\quad f_{12}=(x_1-x_2)\left(x_1+x_2-6x_3+x_4\right)$,
2. $x_1\leftrightarrow x_3$ $\quad f_{13}=(x_1-x_3)\left(x_1-6x_2+x_3+5x_4\right)$,
3. $x_1\leftrightarrow x_4$ $\quad f_{14}=\frac{1}{2}(x_1-x_4)\left(3x_1-6x_2+2x_3+3x_4\right)$,
4. $x_2\leftrightarrow x_3$ $\quad f_{23}=4(x_2-x_3)x_4$,
5. $x_2\leftrightarrow x_4$ $\quad f_{24}=\frac{1}{2}(x_2-x_4)\left(x_2-8x_1+14x_3+x_4\right)$,
6. $x_3\leftrightarrow x_4$ $\quad f_{34}=\frac{1}{2}(x_3-x_4)\left(6x_2-8x_1+x_3+x_4\right)$.

Розв'язуємо задачу 1: $\max(f_{12}^1,f_{13}^1,f_{14}^1,f_{23}^1,f_{24}^1,f_{34}^1)$, де

$$f_{12}^1=(x_1-x_2),\ f_{13}^1=(x_1-x_3),\ f_{14}^1=(x_1-x_4),$$
$$f_{23}^1=(x_2-x_3),\ f_{24}^1=(x_2-x_4),\ f_{34}^1=(x_3-x_4).$$

Перший опорний розв'язок рівний: $(4,3,2,1)$.

Задача 2. Знайти $\max(f_{12}^2,f_{13}^2,f_{14}^2,f_{23}^2,f_{24}^2,f_{34}^2)$, де

$$f_{12}^2=\left(x_1+x_2-6x_3+x_4\right),\ f_{13}^2=\left(x_1-6x_2+x_3+5x_4\right),\ f_{14}^2=\left(3x_1-6x_2+2x_3+3x_4\right),$$
$$f_{23}^2>0,\ f_{24}^2=\left(x_2-8x_1+14x_3+x_4\right),\ f_{34}^2=\left(6x_2-8x_1+x_3+x_4\right).$$

Для знаходження максимуму функції необхідно скористатися 1-м опорним розв'язком задачі. Підставляємо перший опорний розв'язок $(4,3,2,1)$ у $f_{12}^2=\left(x_1+x_2-6x_3+x_4\right)$, $f_{12}^2<0$, щоб функція зростала, оскільки необхідно

знайти максимум функції, необхідно, щоб $x_1 < x_2$, отже другий опорний розв'язок $(3,4,2,1)$. Підставляємо 2-й опорний розв'язок $(3,4,2,1)$ у $f_{13}^2 = (x_1 - 6x_2 + x_3 + 5x_4)$. $f_{13}^2 < 0$, щоб функція зростала, необхідно, щоб $x_1 < x_3$, отже другий опорний розв'язок $(2,4,3,1)$. Аналогічно попереднім міркуванням, $f_{14}^2 < 0$, $x_1 < x_4$, третій опорний розв'язок $(1,4,3,2)$. Оскільки, $f_{24}^2 > 0$ і $f_{34}^2 > 0$, то останній опорний розв'язок $(1,4,3,2)$ буде оптимальним.

Отже, максимальне значення квадратичної функції

$$F(x) = \frac{1}{2}(2x_1 - x_2 - x_3 + x_4)^2 + (x_2 + x_3)^2 - \frac{1}{2}(x_3 - x_2 + 2x_4)^2 + 2x_4^2$$

на множині перестановок $P(A)$ буде в точці $(1,4,3,2)$.

5.3. Оптимізація дробово-лінійної функції на комбінаторних конфігураціях

Дробово-лінійна функція — це функція, що характеризується відношенням двох лінійних форм. Як відомо, такі функції застосовуються у низці прикладних задач оптимізації деяких відносних показників якості, таких як собівартість, рентабельність, продуктивність, трудомісткість і т. д. Моделі, що використовують вказані критерії, відображають тенденції постійного зниження рівня собівартості з розрахунку на одиницю продукції і підвищення якісних показників виробництва при збільшенні масштабів виробництва.

Слід зазначити, що множини — область допустимих розв'язків задачі з дробово-лінійною функцією у багатьох задачах мають властивості комбінаторних конфігурацій: перестановок, розміщень, сполучень та ін.

5.3.1. Постановка задачі з дробово-лінійними функціями критеріїв на графах

Екстремальні задачі на комбінаторних конфігураціях являють собою оптимізацію деякої функції $F(x)$ на комбінаторних конфігураціях, що

визначаються множиною $A = \{\pi_i\} = \{(a_1, a_2, \ldots, a_r)\}$, де $\pi_i = a_1, a_2, \ldots, a_r$, за допомогою якої формуються перестановки, сполучення, розміщення, різні послідовності.

Екстремальні комбінаторні задачі з лінійною цільовою функцією тісно пов'язані з методами лінійного програмування. Якщо ж функція дробово-лінійна, то як правило застосовуються методи лінеаризації функції — тобто зведення її до лінійної форми. Далі розглядається задача знаходження екстремуму цієї функції на опуклій оболонці заданих точок — на опуклому многограннику. Знову екстремум лінійної форми на многограннику досягається в одній з вершин, які входять у множину цих елементів. Особливістю комбінаторних задач при такому зведенні залишиться те, що при знаходженні розв'язків необхідно визначити точки з цілочисловими координатами. Для розв'язання екстремальних задач з дробово-лінійною функцією це не завжди є доцільним і виправданим. Навіть для лінійної цільової функції використання методів лінійного програмування не завжди адекватно дає можливість інтерпретувати розв'язки задачі. Але враховуючи особливості екстремальних задач на комбінаторних конфігураціях, для цієї задачі з дробово-лінійною функцією цілі є також доцільним розглянути підзадачу визначення гамільтонового шляху, який відображає зміну значення дробово-лінійного функціонала на комбінаторних конфігураціях.

Розглядається задача на множині перестановок. Як відомо, переставний многогранник можна представити у вигляді графа, який описаний в роботі [5]. У цьому графі вершинами слугує множина всіх перестановок P_n, а дві вершини $p_1, p_2 \in P_n$ утворюють дугу $\vec{p_1 p_2}$, якщо $f(p_1) \geq f(p_2)$, і якщо перестановка p_2 одержана з p_1 за допомогою транспозиції двох елементів.

Тоді розглянемо таку задачу: знайти множину перестановок, у яких значення цільової функції рівно заданому значенню, тобто

$$x^* = \underset{x \in P_n}{arg}\, f(x), \text{ де } f(x^*) = y.$$

У цьому випадку функція представляється в дробово-лінійному вигляді, як відношення двох лінійних форм, що значно ускладнює задачу. Тоді вищесформульована задача зводиться до знаходження множини точок — перестановок, у яких досягається задане значення дробово-лінійного функціоналу:

$$x^* = \underset{x \in P_n}{arg}\, F(x) = \underset{x \in P_n}{arg}\, \frac{f(x)}{g(x)}, \tag{5.9}$$

де $y = F(x)$.

Також має сенс розглядати аналогічну задачу, де не завжди існують перестановки, у яких цільова функція приймає задане значення. Тоді вищесформульована проблема сформулюється таким чином: визначити множину пар перестановок $(\underline{x}, \overline{x})$, для яких при заданому y

$$\overline{x} = \underset{F(x)>y}{arg\,min}\, F(x) = \underset{F(x)>y}{arg\,min}\, \frac{f(x)}{g(x)}, \quad \underline{x} = \underset{F(x)<y}{arg\,max}\, F(x) = \underset{F(x)<y}{arg\,max}\, \frac{f(x)}{g(x)}. \tag{5.10}$$

5.3.2. Властивості області допустимих значень дробово-лінійних функцій на графах

Зазначимо деякі властивості дробово-лінійної функції, які будуть корисні для подальшого розв'язання задачі.

Дробово-лінійна функція $F(x) = (\langle c, x\rangle + c_0) / (\langle d, x\rangle + d_0)$ не є ні увігнутою, ні опуклою. Поверхні рівня будь-якої функції $F(x)$, тобто множини $L_\alpha = \{x \in R^n \,|\, F(x) = \alpha\}$ є гіперплощинами.

Відомо, що будь-який локальний мінімум задачі дробово-лінійного програмування є в той же час глобальним, і якщо оптимальний розв'язок скінченний, то існує крайня точка многогранника M — області допустимих розв'язків, яка є оптимальною. Це твердження виконується, якщо чисельник і знаменник дробово-лінійної функції не перетворюються одночасно в нуль $\forall x \in X$.

Припустимо, що $\langle d,x\rangle + d_0 > 0 \; \forall x \in M$. Відомо, що на будь-якому прямолінійному відрізку, який належить многограннику *M*, дробово-лінійна функція $F(x)$ змінюється монотонно.

Теорема 5.1. Дробово-лінійна функція $F(x)$, досягає мінімуму (максимуму) тільки у вершинах многогранника *M*. Якщо мінімум (максимум) досягається у декількох крайніх точках, то він досягається і на їх опуклій оболонці.

Визначення 5.3. Неперервна функція $F(x)$ є квазіопуклою функцією на опуклій множині *M*, якщо виконується будь-яка з таких еквівалентних умов:

(a) множина $\{x \in R^m \mid F(x) \le q, x \in M\}$ — опукла для усіх q, $M \subset R^n$;

(b) $x_1, x_2 \in M, F(x_2) < F(x_1) \Rightarrow F(\lambda x_2 + (1-\lambda)x_1) \le F(x_1), \; 0 \le \lambda \le 1.$

Визначення 5.4. Функція $F(x)$ є строго квазіопуклою на множині *M*, якщо в умові (*b*) фігурують строгі нерівності.

Теорема 5.2. Будь-який локальний мінімум у строго квазіопуклій функції є глобальним.

Оскільки множина $\left\{x \in R^n \middle| \left(\langle c\ ,x\rangle + c_0\right) \middle/ \langle d\ ,x\rangle + d_0 \le q, x \in M\right\}$ опукла для усіх значень *q*, то функція $F(x) = \left(\langle c,x\rangle + c_0\right) / \left(\langle d,x\rangle + d_0\right)$ є квазіопукла на множині *M*. Неважко показати, що функція $\langle c,x\rangle / \langle d,x\rangle$ є строго квазіопуклою на *M*.

Очевидно, що функція $F(x)$ є квазіувігнутою на опуклій множині *S*, якщо функція $(-F(x))$ квазіопукла. Якщо зробити заміну: $F_i(x)$ на $-F_i(x)$, ту умову (*b*) для квазіувігнутих функцій можна записати в такому вигляді:

$$F_i(x_2) > F_i(x_1) \Rightarrow F_i(\lambda x_1 + (1-\lambda)x_2) \ge F_i(x_1),$$

$F_i(\lambda x_1 + (1-\lambda)x_2) \ge F_i(x_1)$, або $F_i(\lambda x_1 + (1-\lambda)x_2) \ge min\left[F_i(x_1), F_i(x_2)\right]$,

для будь-яких $x_1, x_2 \in S$ і $0 \le \lambda \le 1$. Для увігнутих і квазіувігнутих функцій з точністю до зміни знаків зберігаються властивості опуклих і квазіопуклих функцій.

5.3.3. Підхід до розв'язання екстремальної задачі з дробово-лінійним функціоналом на перестановках

Розглянемо дробово-лінійну функцію $F(x)$, де чисельник $f(x)$ і знаменник $g(x)$ — дві лінійні функції, у яких коефіцієнти визначаються за правилом арифметичної прогресії:

$$c_i = c_1 + \Delta(i-1),\ d_i = d_1 + \delta(i-1). \tag{5.11}$$

Згідно з цим позначенням чисельник і знаменник дробово-лінійної функції для цієї перестановки p можна представити відповідно: $f(x) = (\vec{c}, x(p))$, де $x(p)$ — вектор змінних $(x_{p(1)}, x_{p(2)}, \ldots, x_{p(n)})$, а $g(x) = (\vec{d}, x(p))$, де $p \in P_n$ — множина перестановок, а множина $(x_1, x_2, \ldots, x_n)$, де $x_i \in N$, тобто перестановці множини $(1, 2, \ldots, n)$ ставиться у відповідність вектор $p = (x_{p(1)}, x_{p(2)}, \ldots, x_{p(n)})$.

Розглянемо значення дробово-лінійної функції у двох різних π, σ перестановках, які відповідають номерам p_1, p_2, та порівняємо їх:

$$\frac{(\overline{c}, x(\pi))}{(\overline{d}, x(\pi))} - \frac{(\overline{c}, x(\sigma))}{(\overline{d}, x(\sigma))}.$$

Визначимо наступне співвідношення, у вигляді коефіцієнта $\rho(\pi, \sigma)$, що характеризуватиме деяку відмінність між значеннями функцій у сусідніх перестановках π, σ, що відрізняються одна від одної однією транспозицією:

$$\rho(p_1, p_2) = (p_1, \overline{c})(p_2, \overline{d}) - (p_2, \overline{c})(p_1, \overline{d}).$$

Тоді, очевидно що, якщо $\rho(p_1, p_2) \geq 0$, то $F(p_1) \geq F(p_2)$.

Розглянемо далі такі міркування, підставивши значення змінних у чисельник і знаменник функції:

$$\overline{c} \cdot x(\pi) = c_1 x_{\pi(1)} + c_2 x_{\pi(2)} + \ldots + c_n x_{\pi(n)} = c_1 \sum_{i=1}^{n} x_i + \Delta(p_0 x(\pi)),$$

$p_0 = (0, 1, \ldots, n-1)$, $\sum_{i=1}^{n} x_i = S$, для знаменника аналогічно.

Визначимо співвідношення різниці:

$$\frac{c_1S+\Delta\left(p_0\cdot x(\pi)\right)}{d_1S+\delta\left(p_0\cdot x(\pi)\right)}-\frac{c_1S+\Delta\left(p_0\cdot x(\sigma)\right)}{d_1S+\delta\left(p_0\cdot x(\sigma)\right)},$$

зробимо математичні перетворення

$$c_1d_1S^2+c_1\delta S\left(p_0x(\sigma)\right)+d_1\Delta S\left(p_0x(\pi)\right)-\Delta\delta\left[\left(p_0\left(x(\pi)\right)\right)\cdot\left(p_0\left(x(\sigma)\right)\right)\right]-$$

$$-c_1d_1S^2-c_1\delta S\left(p_0x(\pi)\right)-d_1\Delta S\left(p_0x(\sigma)\right)-\Delta\delta\left[\left(p_0\left(x(\pi)\right)\right)\cdot\left(p_0\left(x(\sigma)\right)\right)\right].$$

Зведемо подібні доданки, одержимо

$$S\left[c_1\delta-d_1\Delta\right]\left(p_0x(\sigma)\right)+S\left[c_1\delta-d_1\Delta\right]\left(p_0x(\pi)\right);$$

$$S\left[c_1\delta-d_1\Delta\right]\left(p_0x(\sigma)-\left(p_0x(\pi)\right)\right)=S\left[c_1\delta-d_1\Delta\right]p_0\left(x(\sigma)-x(\pi)\right).$$

Розглянемо дві перестановки:

$$\pi=\left(x_{\pi(1)},x_{\pi(2)},\ldots,x_{\pi(j)},\ldots,x_{\pi(k)},\ldots\right),\ \sigma=\left(x_{\pi(1)},x_{\pi(2)},\ldots,x_{\pi(k)},\ldots,x_{\pi(j)},\ldots\right),$$

$$(j-1)\left[-x_{\pi(j)}+x_{\pi(k)}\right]+(k-1)\left[x_{\pi(k)}-x_{\pi(j)}\right]=(k-j)\left[x_{\pi(k)}-x_{\pi(j)}\right].$$

Далі приведемо міркування для $n=3$, що визначає кількість елементів перестановки в множині P_n, складених із множини $A(a_1,a_2,a_3)$, оскільки їх же можна адаптувати на більш загальний випадок.

Для прикладу розглянемо дві перестановки $p_1=(1,2,3)$, $p_2=(1,3,2)$, підставивши їх у цільову функцію $F(x)$ і розглянемо знак різниці між ними:

$$\frac{c_1x_1+c_2x_2+c_3x_3}{d_1x_1+d_2x_2+d_3x_3}\sim\frac{c_1x_1+c_2x_3+c_3x_2}{d_1x_1+d_2x_3+d_3x_2}. \tag{5.12}$$

Помітимо, що знак різниці визначається знаком чисельника, а знаменник не впливає на знак, тому розглянемо співвідношення чисельника (5.12).

Після елементарних математичних перетворень над пропорціями, одержимо такий вираз:

$$x_2^2\left(c_2d_3-c_3d_2\right)+x_3^2\left(c_3d_2-c_2d_3\right)+x_1x_2\left(c_1d_3-c_3d_1+c_2d_1-c_1d_2\right)+$$
$$+x_1x_3\left(c_1d_2-c_2d_1+c_3d_1-c_1d_3\right). \tag{5.13}$$

Введемо такі позначення:

$$\begin{vmatrix} c_1, c_2 \\ d_1, d_2 \end{vmatrix} = m_{12}; \quad \begin{vmatrix} c_1, c_3 \\ d_1, d_3 \end{vmatrix} = m_{13}; \quad \begin{vmatrix} c_2, c_3 \\ d_2, d_3 \end{vmatrix} = m_{23}. \tag{5.14}$$

Підставляючи позначення (5.14) в (5.13), одержимо такі вирази:

$$x_2^2 m_{23} - x_3^2 m_{23} = m_{23}\left(x_2^2 - x_3^2\right),$$

$$x_1 x_2 \left(c_1 d_3 - c_3 d_1 + c_2 d_1 - c_1 d_2\right) = x_1 x_2 \left(m_{13} - m_{12}\right),$$

$$x_1 x_3 \left(c_1 d_2 - c_2 d_1 + c_3 d_1 - c_1 d_3\right) = x_1 x_3 \left(m_{12} - m_{13}\right)$$

і після зведення подібних доданків, маємо:

$$m_{12}\left(x_1 x_3 - x_1 x_2\right) + m_{13}\left(x_1 x_2 - x_1 x_3\right) + m_{23}\left(x_2^2 - x_3^2\right).$$

Введемо коефіцієнт $\rho\left(p_i, p_j\right)$, який визначає знак різниці між заданими перестановками p_i, p_j.

Визначення 5.5. Мінор матриці, утвореної двома рядками коефіцієнтів, заданими згідно з формулою (5.11), цільової дробово-лінійної функції $F(x)$,

$$m_{ij} = \begin{vmatrix} c_i & c_j \\ d_i & d_j \end{vmatrix}$$

називається мінором цільової функції, що визначає знак між двома перестановками p_i, p_j, у яких досягається значення цієї цільової функції.

Тоді, враховуючи (5.13) і правило знаходження визначника між перестановками p_1, p_2, знак різниці визначається таким чином:

$$\rho\left(p_1, p_2\right) = \begin{vmatrix} 1\,2\,3 \\ 1\,3\,2 \end{vmatrix} = m_{12} - m_{13} - 5m_{23}$$

і залежить від мінору другого порядку m_{ij}.

Визначення 5.6. Назвемо коефіцієнти при мінорах m_{12}, m_{13}, m_{23} відповідно λ_{12}, λ_{13}, λ_{23}, тоді $\vec{\lambda}(i, j) = \left(\lambda_{12}, \lambda_{13}, \lambda_{23}\right)$.

Враховуючи, що для $n = 3$ для лінійної функції $f(x)$ структурний граф переставного многогранника M матиме такий вигляд (рис. 5.5):

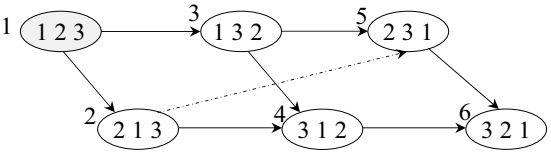


Рис. 5.5. Структурований граф переставного многогранника $n=3$

Визначимо відстані між точками:

$$\rho(1,3)=(1,-1,-5),\ \rho(1,2)=(-3,-3,3),\ \rho(2,4)=(-1,-5,-1),$$
$$\rho(3,4)=(-8,-4,4),\ \rho(3,5)=(-3,-3,-3),\ \rho(4,6)=(3,-3,-3),$$
$$\rho(5,6)=(-5,-1,1),\ \rho(1,4)=(-5,-7,1),\ \rho(1,6)=(-4,-8,-4),$$
$$\rho(2,3)=(5,1,-7).$$

Враховуючи, що коефіцієнти цільової функції визначені згідно з формулами

$$c_i=c_1+\Delta(i-1),\ d_i=d_1+\delta(i-1),$$

зробимо заміну у виразах (5.13); тоді маємо такі вирази:

$$m_{12}=\begin{vmatrix}c_1, c_1+\Delta\\ d_1, d_1+\delta\end{vmatrix}=\delta c_1-\Delta d_1;$$

$$m_{13}=\begin{vmatrix}c_1, c_1+2\Delta\\ d_1, d_1+2\delta\end{vmatrix}=2(\delta c_1-\Delta d_1)=2m_{12};$$

$$m_{23}=\begin{vmatrix}c_1+\Delta, c_1+2\Delta\\ d_1+\delta, d_1+2\delta\end{vmatrix}=\begin{vmatrix}c_1+\Delta, \Delta\\ d_1+\delta, \delta\end{vmatrix}=\begin{vmatrix}c_1, \Delta\\ d_1, \delta\end{vmatrix}=m_{12}. \qquad (5.15)$$

Підставляючи (5.15) в значення відстаней між точками-перестановками, одержимо:

$$\rho(1,3)=-6m_{11},\ \rho(1,2)=-6m_{11},\ \rho(2,4)=-12m_{11},\ \rho(3,4)=-12m_{11},$$
$$\rho(3,5)=-6m_{11},\ \rho(4,6)=-6m_{11},\ \rho(5,6)=-6m_{11}.$$

На підставі зроблених розрахунків коефіцієнтів можна побудувати гамільтонів шлях усередині кожної шестірки, елементів перестановки і прослідити зміни значень цільової функції у вершинах графа та на кожному з підграфів, так як і для лінійної функції.

На підставі вищевисловлених міркувань для переставного многогранника $n=3,$ сформулюємо деякі значимі факти для довільного значення n.

Лема 5.2. Для $n \geq 3$ і $(i \neq j)$ справедлива наступна рівність:

$$m_{ij} = \begin{vmatrix} c_1 + (j-i)\Delta \\ d_1 + (j-i)\delta \end{vmatrix} = |i-j| \begin{vmatrix} c_1 & \Delta \\ d_1 & \delta \end{vmatrix}, \quad m_{ij} = |i-j| m_{12}.$$

Доведення леми виходить з вищевисловлених міркувань.

Тоді граф перестановок при $n=4$ для значень дробово-лінійної функції має такий вигляд (рис. 5.6).

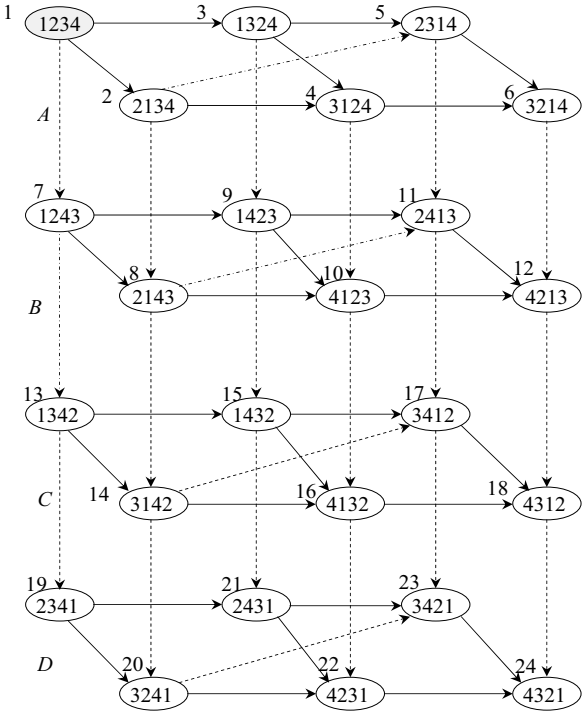


Рис. 5.6. Представлення графа перестановок $G(P_n)$ для $n=4$

Теорема 5.3. Різниця між двома перестановками $\rho(p_i, p_j),\ (i \neq j)$ для довільних n визначається за формулою

$$\rho\left(p_i, p_j\right) = \sum_{j=i+1}^{n} \sum_{i=1}^{n-1} \lambda_{ij} m_{ij}, \tag{5.16}$$

де λ_{ij} дорівнює мінору матриці $2 \times n$, утвореної двома перестановками p_i, p_j.

Доведення: проводимо методом за індукцією:

1) для $k = 1$ цей вираз має місце;

2) для k цей вираз виконується на підставі вищевисловлених міркувань;

3) для $k+1$ доведемо цей факт. Для цього розглянемо p_t, p_s, які утворені однією транспозицією, тобто відрізняються порядком слідування тільки однієї координати: $p_t = \left(i_1, i_2, \ldots, i_t, i_s, \ldots, i_k, i_{k+1}\right)$, $p_s = \left(i_1, i_2, \ldots, i_s, i_t, \ldots, i_k, i_{k+1}\right)$.

Складемо матрицю їх коефіцієнтів, яка виглядає так:

$$\begin{bmatrix} i_1, i_2, \ldots, i_t, i_s, \ldots, i_k, i_{k+1} \\ \underbrace{i_1, i_2, \ldots}_{t-1}, i_t, i_s, \underbrace{\ldots, i_k, i_{k+1}}_{k+1-s} \end{bmatrix}. \tag{5.17}$$

Відповідно, щоб знайти різницю між двома перестановками $\rho\left(p_t, p_s\right)$, необхідно використати формулу (5.12). Далі знак різниці визначити перетвореннями формул (5.15). Формула (5.16) випливає із означення мінорів і коефіцієнтів при мінорах.

Теорема 5.4. Граф перестановок $\tilde{G}\left(P_n\right)$ для дробово-лінійної функції $F\left(x\right)$, коефіцієнти якої визначені згідно з (5.10), збігається з графом перестановок для лінійної функції $G\left(P_n\right)$, з точністю до орієнтації.

Доведення: випливає з леми 5.2.

Наслідок 5.1. Екстремальні значення функції $F\left(x\right)$, в якій коефіцієнти цільової функції визначаються за формулами (5.10), досягається у крайніх точках.

5.4. Оптимізація лінійної та квадратичної функцій на комбінаторних конфігураціях з обмеженнями

Розглядається загальна схема для розв'язування екстремальних комбінаторних задач з додатковими умовами. Слід зазначити, що додаткові

умови ускладнюють процес розв'язку задачі, але зменшують область допустимих розв'язків, звужуючи тим самим множину допустимих розв'язків.

5.4.1. Постановка екстремальних задач з обмеженнями

Розглянемо екстремальну задачу комбінаторної оптимізації як обчислювальну проблему, у якій задана множина альтернатив $X = \{x\}$, цільова функція $f(x): X \to R$, і потрібно знайти альтернативу $x^0 \in X$, на якій ця цільова функція приймає екстремальне значення: $f(x^0) = \underset{x \in X}{extr} f(X)$, $extr \in \{\min, \max\}$. Для задач оптимізації альтернативи $x \in X$ звичайно називають допустимими розв'язками, x^0 — оптимальний розв'язок, $X = \{x\}$ — множиною допустимих розв'язків.

У цьому випадку розглядається X — деяка задана комбінаторна конфігурація. Задача може містити також додаткові лінійні обмеження, які утворять опуклу многогранну множину $D \subset R^n$ вигляду: $D = \{x \in R^n \mid Gx \le h\}$, де $G \in R^{m \times n}$, $h \in R^m$. Запишемо лінійні обмеження у вигляді лінійних нерівностей:

$$G \cdot x = \sum_{j=1}^{n} g_{ij} x_j \ge h_i;\ i \in N_m;\ j \in N_n. \tag{5.18}$$

Конкретизуємо задачу і розглянемо її на комбінаторній конфігурації перестановок. Розглянемо перестановки з множини $A = \{1, 2, ..., n\}$. Кількість елементів множини перестановки $P_n(A)$ дорівнює $n!$.

Послідовності перестановок, згідно з методом їх генерування з попередніх розділів інтерпретуються як граф G_n, вершини якого відповідають усім точкам множини перестановок $P_n(A)$.

Для однокритеріальної задачі $F(x) = f(x)$ без додаткових обмежень максимальне значення лінійної функції $f(x)$ при упорядкуванні коефіцієнтів за

зростанням на графі перестановок $G(P_n)$ досягається у перестановці $(1,2,...,n)$, а мінімальне — у перестановці $(n,n-1,...,2,1)$. Враховуючи умову (5.18), сформулюємо підзадачі, що необхідно розв'язати для визначення підмножини допустимих перестановок: визначити множину зв'язних пар перестановок $(\underline{x},\overline{x})$, для яких при заданому $h_i, i \in N_m$ має місце

$$\overline{x} = \arg\min_{Gx \geq h_i} G(x), \tag{5.19}$$

$$\underline{x} = \arg\max_{Gx < h_i} G(x). \tag{5.20}$$

Загальна схема цього алгоритму полягає в наступному.

Початковий етап: Початкова множина перестановок M_0 замінюється на базову M за допомогою вихідної перестановки u, яка нормалізує цільову функцію $f(x)$. За допомогою цієї самої перестановки переводимо множину перестановок кожної обмежуючої функції в базову (спільну для всіх функцій). Тепер для кожної i-ї обмежуючої функції g_i робимо такі операції:

1) нормалізуємо функцію за допомогою перестановки u_i, отримуючи індивідуальну множину перестановок M_i; тоді для кожної обмежуючої функції граф перестановок $G(P_n)$ має стандартний вигляд;

2) розв'язуючи задачу локалізації для $h_i, i \in N_m$, отримуємо допустиму множину перестановок для g_i;

3) за допомогою перестановки u_i^{-1} визначаємо ту саму множину в базовій множині перестановок M.

4) знаходимо на цій множині $extr$ функції g_i.

5) після усіх цих операцій відносно кожної обмежуючої функції знаходимо розв'язок задачі як $\min\{extr\ g_i\}$.

6) за допомогою перестановки u_i^{-1} знаходимо розв'язок задачі в початковій множині перестановок M_0.

Перед тим, як розглянути приклад, детальніше розберемо в алгоритмі пункти 2 і 3. Підграф графа $G(P_n)$, у якого зафіксовано останній елемент i, назвемо гранню графа $G(P_n)$ і позначимо G_i. Усі грані зображатимемо у вигляді ребра, інцидентні вершини якого є початкова та кінцева вершини відповідного підграфа. В результаті отримаємо структурну схему графа $G(P_n)$, яка має вигляд драбини (рис. 5.7):

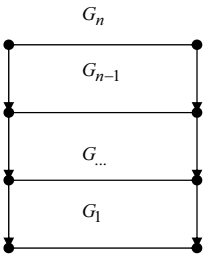


Рис. 5.7. Структурна схема графа $G(P_n)$

Аналогічно робиться структурна схема і підграфів G_i, тільки в них будуть фіксовані 2 індекси (і так далі за індукцією). Виконуючи пункт 2 в алгоритмі, отримаємо спочатку структурну схему для обмежуючої функції $G_i(g_i)$, $(i=1,2,\ldots,m)$. Подрібнюючи структурні схеми, отримаємо допустиму множину перестановок для g_i у вигляді підграфів з різною кількістю індексів. Тепер для виконання пункту 3 необхідно подіяти оберненою перестановкою u_i^{-1} на індекси цих підграфів і отримаємо шукану кількість таких самих підграфів з індексами, але уже в базовій множині.

Розглянемо цей алгоритм на прикладі: дано функцію $f(x)=4x_1+2x_2+7x_3+9x_4$, елементи множини перестановок для $P_4=\{1,2,3,4\}$; задані додаткові обмеження на перестановки, що визначають область допустимих значень цільової функції: $g_1=5x_1+6x_2+10x_3+1x_4\le 60$, $g_2=4x_1+6x_2+8x_3+9x_4\ge 63$.

Знайти: екстремум цільової функції — максимум.

Розв'язання.

Початковий етап. Нормалізуємо задану цільову функцію $f(x)$ у порядку зростання за допомогою перестановки $u = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 3 & 4 \end{pmatrix}$. Отримаємо цільову функцію $f(x) = 2x_1 + 4x_2 + 7x_3 + 9x_4$. Граф перестановок для нормалізованих функцій має вигляд, представлений на рис. 5.8.

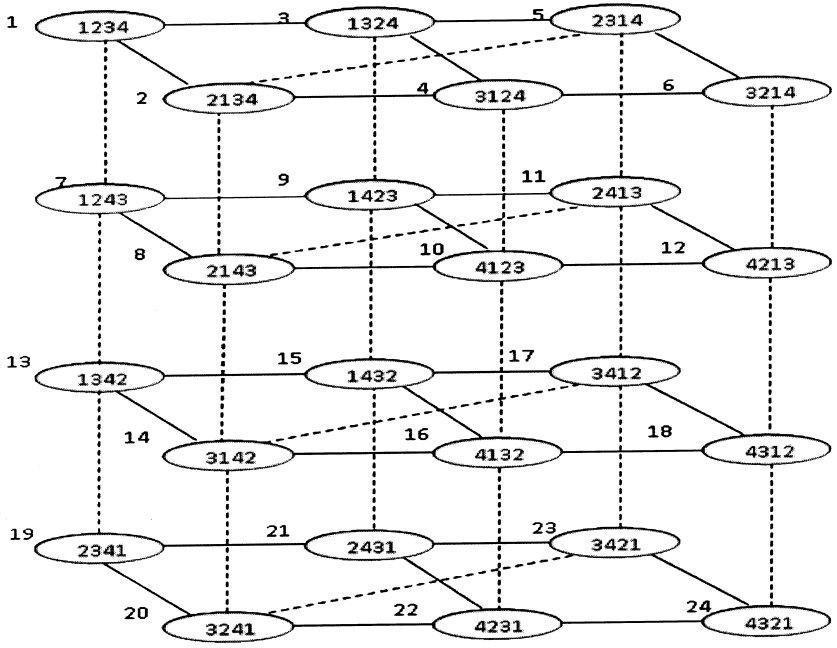


Рис. 5.8. Стандартний вигляд графа перестановок

Нормалізуємо також обмежуючі функції $g_1 = 6x_1 + 5x_2 + 10x_3 + 1x_4$ та $g_2 = 6x_1 + 4x_2 + 8x_3 + 9x_4$ за допомогою перестановки, що і цільову функцію $u = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 3 & 4 \end{pmatrix}$.

Крок 1. Нормалізуємо g_1 за допомогою перестановки $u_1 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 2 & 4 & 1 \end{pmatrix}$ до вигляду $g_1 = x_1 + 5x_2 + 6x_3 + 10x_4$.

Структурна схема графу G розбивається на структурні схеми підграфів G_1, G_2, G_3, G_4, і знаходиться значення функції в крайніх точках підграфів, які визначають мінімальне та максимальне значення на цих підграфах (рис. 5.9)

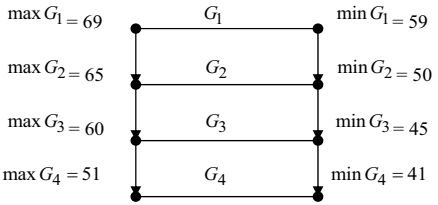


Рис. 5.9. Загальна структурна схема графу для g_1

Розв'язуючи далі задачу про локалізацію, отримаємо в G_1 дві вершини, які не задовольняють обмеженням, це $4 = (3124)$ та $6 = (3214)$. У G_2 це такі вершини 10, 11, 12, в G_3 усі, крім 13, а в G_4 усі вершини задовольняють обмеженням. У сумі маємо множину вершин (4, 6, 10, 11, 12, 13, 19, 20, 21, 22, 23, 24). За допомогою оберненої перестановки $u_1^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 2 & 1 & 3 \end{pmatrix}$ з цієї множини в базовій множині отримаємо такі перестановки (8, 7, 2, 15, 1, 24, 18, 12, 17, 6, 11, 5), які не задовольняють обмеженням. Без сумніву, максимальне значення цільова функція приймає у точці $3 = (1324)$, яке дорівнює 64. Зауважимо, що підграфу G_4, вершини якого задовольняють умовам функції g_1, відповідає в базовій множині підграф G_3, тому що в підстановці $u_i^{-1}4$ переходить у 3.

Крок 2. Нормалізуємо g_2 за допомогою перестановки $u_2 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 3 & 4 \end{pmatrix}$ до вигляду $g_2 = 4x_1 + 6x_2 + 8x_3 + 9x_4$.

Структурна схема графу G розбивається на структурні схеми підграфів G_1, G_2, G_3, G_4, і визначається значення функції в крайніх точках підграфів, які визначають мінімальне та максимальне значення на підграфах

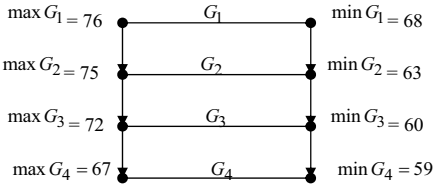


Рис. 5.10. Загальна структурна схема графу для g_2

Усі вершини підграфів G_1 та G_2, що зображені на структурній схемі графа (рис. 5.10), задовольняють обмеженням. У G_3 їм не задовольняють дві вершини 17 та 18. В G_4 обмеженням не задовольняють три вершини 22, 23 та 24. За допомогою оберненої перестановки $u_2^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 3 & 4 \end{pmatrix}$ з цієї множини в базовій множині отримаємо такі самі перестановки. Оскільки вершина 2 входить в обмеження і для функції g_2, то розв'язок задачі не зміниться — максимальне значення цільова функція приймає у точці $3 = (1324)$, яке дорівнює 64.

Залишається повернутися у початкову множину перестановок: максимальний розв'язок задачі буде на перестановці (1324).

5.4.2. Розв'язування екстремальних задач на конфігурації сполучень

Для конфігурації сполучень елементи можна зобразити у вигляді дерева, оскільки елементи сполучень згідно з методом генерування можна розмістити у вигляді неорієнтованого графу, що не має циклів, тобто такого, в якому кожна пара вершин з'єднана одним простим шляхом (ланцюгом).

Досить цікавими є задачі на множині сполучень, в яких необхідно оперувати із симетричними функціями типу $\alpha x_1 x_2 + \alpha x_2 x_3 + \ldots + \alpha x_{n-1} x_n$. Оскільки такі функції є симетричними, то немає потреби в нормалізації функції. Тоді загальна схема алгоритму для екстремальних задач на сполученнях полягає в такому:

Початковий етап. Вибирається базова M множина за допомогою вихідної перестановки u.

Тепер для кожної i-ї обмежуючої функції g_i зробимо такі операції:

1) визначаємо для кожної обмежуючої функції дерево $G(P_n)$, що має стандартний вигляд, отримуючи індивідуальну множину сполучень M_i (рис. 5.11);

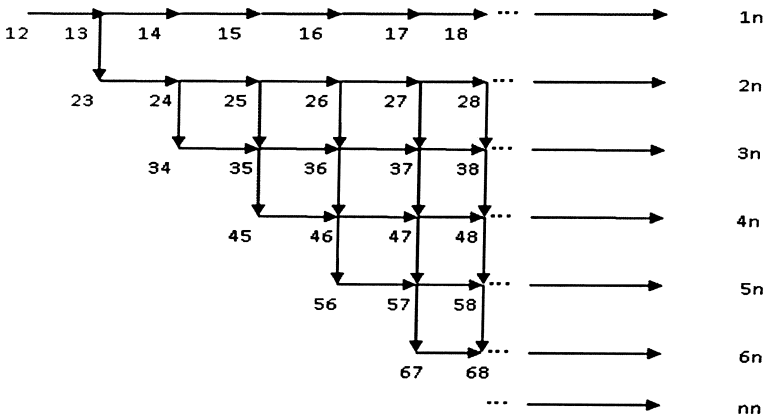


Рис. 5.11. Стандартний вигляд дерева сполучень

2) розв'язуючи задачу локалізації для $h_i, i \in N_m$, отримуємо допустиму множину сполучень для g_i;

3) за допомогою перестановки u_i^{-1} визначаємо ту саму множину в базовій множині сполучень M;

4) знаходимо на цій множині $extr$ функції g_i;

5) після усіх цих операцій відносно кожної обмежуючої функції знаходимо розв'язок задачі як $\min\{extr\ g_i\}$;

6) за допомогою перестановки u^{-1} знаходимо розв'язок задачі в початковій множині сполучень M_0.

Розглянемо цей алгоритм на прикладі.

Дано функцію $f(x) = 6x_1x_2 + 6x_1x_3 + 6x_2x_3$, елементи множини, що формують множину сполучень $A_i(1\ 2\ 3\ 4\ 5\ 6)$; задані додаткові обмеження, які визначають область допустимих значень цільової функції:

$g_1 = 4x_1x_2 + 4x_2x_3 + 4x_1x_3 \geq 165$, $g_2 = 5x_1x_2 + 5x_2x_3 + 5x_1x_3 \leq 220$.

Знайти: екстремум цільової функції — максимум.

Початковий етап. Нормалізуємо задану цільову функцію $f(x)$ у порядку зростання за перестановкою $u = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix}$; отримаємо цільову функцію $f(x) = 6x_1x_2 + 6x_1x_3 + 6x_2x_3$, а також обмежуючі функції $g_1 = 4x_1x_2 + 4x_2x_3 + 4x_1x_3$, $g_2 = 5x_1x_2 + 5x_2x_3 + 5x_1x_3$.

Розглянемо конфігурацію сполучень з 6 по 3, де $C_6^3 = 20$.

Формуємо базове дерево елементів сполучення для цільової функції: вибираємо з впорядкованої за зростанням множини із шести заданих елементів 1,2,3,4,5,6 перші три 1, 2, 3 і для побудови першого піддерева останній елемент замінюємо наступними із заданої множини, маємо: 1,2,4; 1,2,5; 1,2,6. Тоді друге піддерево утворюється шляхом заміни в кожному з попередніх елементів, крім першого другої компоненти на наступний елемент із заданої множини маємо 1, 3, 4; 1, 3, 5; 1, 3, 6. Маємо базове дерево (рис. 5.12).

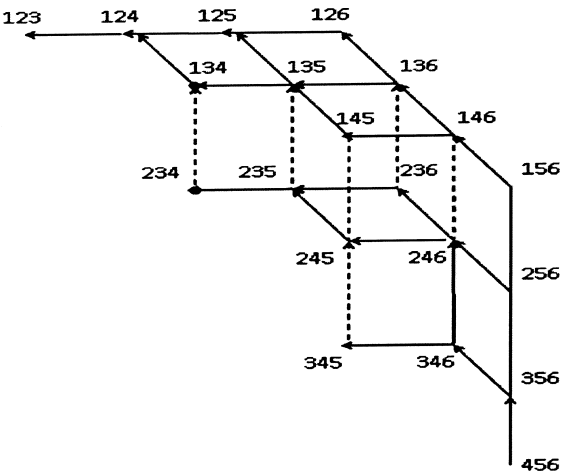


Рис. 5.12. Стандартний вигляд дерева конфігурації сполучень

На основі базового дерева будуємо схему структурного дерева рішень (рис. 5.13).

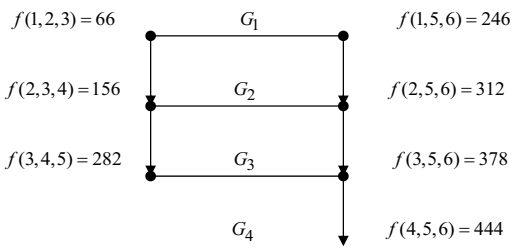


Рис. 5.13. Структурна схема піддерева сполучень для функції цілі $f(x)$

Крок 1. Нормалізуємо g_1 за допомогою перестановки $u_1 = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix}$ до вигляду $g_1 = 4x_1x_2 + 4x_2x_3 + 4x_1x_3$.

Розглядаємо структурну схему дерева конфігурації сполучень, що можна представити таким чином (рис. 5.14):

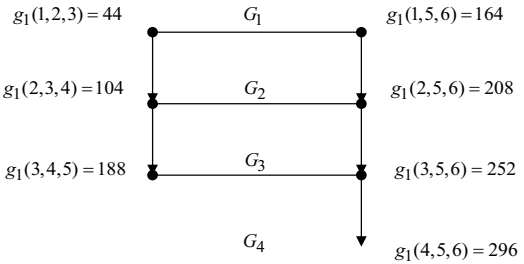


Рис. 5.14. Структурна схема піддерева сполучень для g_1

Структурна схема дерева G розбивається на піддерева G_1, G_2, G_3 і визначається значення функції в крайніх точках піддерев, які визначають мінімальне та максимальне значення на піддеревах (див. рис. 5.7).

Розв'язуючи далі задачу про локалізацію та враховуючи, що для сполучень, які можуть бути розв'язками задачі, необхідно виконання умови $g_1 = 4x_1x_2 + 4x_2x_3 + 4x_1x_3 \geq 165$, згідно з рис. 5.14 розв'язками будуть усі вершини піддерев G_3, G_4. А з множини вершин піддерев G_1 — не задовольняє жодна з вершин, G_2 необхідно розглянути наступні (2 4 6), (2 5 6).

Крок 2. Нормалізуємо g_2 за допомогою перестановки $u_2 = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix}$ до вигляду $g_2 = 5x_1x_2 + 5x_2x_3 + 5x_1x_3$. Структурна схема дерева розбивається на піддерева G_1, G_2, G_3, G_4, для яких визначається значення функції в крайніх вершинах, що задовольняють обмеження $g_2 = 5x_1x_2 + 5x_2x_3 + 5x_1x_3 \leq 220$ (рис. 5.15).

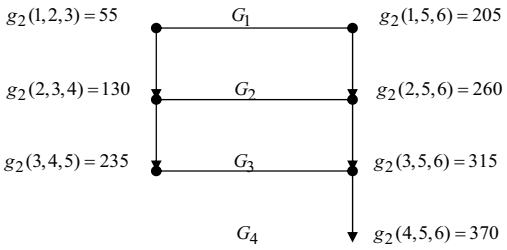


Рис. 5.15. Структурна схема дерева сполучень для g_2

Всі вершини піддерева G_1 задовольняють обмеженням. У G_2 обмеженням задовольняють лише такі точки (2 3 4), (2 3 5), (2 3 6), (2 4 5), (2 4 6). В G_3 і G_4 обмеженням не задовольняють усі вершини.

Шукаємо перетин структурних схем піддерев для обох додаткових обмежень — це вершини піддерева G_2.

Одержимо вершину (2 4 6), що задовольняє обидва обмеження, а функція набуває максимального значення в ній — 264 серед множини визначених точок, що є областю допустимих значень, то вершина (2 4 6) є розв'язком задачі.

Список літератури

1. Донец Г. А., Колечкина Л. Н. Об одном подходе к решению комбинаторной задачи оптимизации на графах. Управляющие системы и машины. 2009. № 4. С. 36–42.
2. Донец Г. А., Колечкина Л. Н. Построение гамильтонова пути в графах перестановочных многогранников. Кибернетика и системный анализ. 2010. № 1. С. 10–16.
3. Ємець О. О., Колєчкіна Л. М. Задачі комбінаторної оптимізації з дробово-лінійними цільовими функціями. Київ: Наукова думка, 2005. 118 с.
4. Донец Г. А., Колечкина Л. Н. Локализация значения линейной функции заданной на перестановках. Радиоэлектроника и информатика. 2009. № 1. С. 76–81.
5. Емеличев В. А., Ковалев М. М., Кравцов М. К. Многогранники, графы, оптимизация. Москва: Наука, 1981. 342 с.

6. МЕТОДИ КОМБІНАТОРНОГО РОЗПІЗНАВАННЯ

Г. П. Донець, В. І. Білецький

Робота присвячена дослідженню та розробці методів комбінаторного розпізнавання предметів з нестандартними властивостями серед маси однотипних. Розглядуються задачі двох типів: обмеженого та необмеженого комбінаторного розпізнавання. Для задачі другого типу описані алгоритми оптимального пошуку двох нестандартних предметів на множині $n \le 22$, отримані деякі результати пошуку трьох і чотирьох таких предметів.

The work is devoted to research and development of combinatorial recognition methods for objects with non-standard properties among objects of the same type. The problems of two types are considered: limited and unlimited combinatorial recognition. For the second type problems algorithms for optimal search of two non-standard objects on the set $n \le 22$ are described, and some results of the search for three and four such objects are obtained.

6.1. Вступ

У теорії комбінаторного розпізнавання розглядаються задачі розпізнавання властивостей окремих предметів із усієї їхньої сукупності за допомогою тестів, експериментів, інших засобів розпізнавання. Такі задачі діляться на два типи.

До першого типу відносяться так звані задачі обмеженого комбінаторного розпізнавання (ОКР), де основною операцією є підрахунок і побудова конкретних комбінаторних конфігурацій на множині заданих чисел. Такі задачі виникають у різних лотереях, банківській сфері, страхових компаніях і відносяться до класу задач обмеженого комбінаторного розпізнавання. Для таких задач стратегія пошуку оптимального розв'язку полягає в розбивці вихідної множини з n елементів на групи з m ($m \le n$) елементів таким чином, щоб потім, виконавши експерименти за допомогою k — вибірок ($k \le m$) у кожній з них, знайти необхідну кількість визначених елементів.

До другого типу відносяться задачі, де потрібно з усієї сукупності однорідних предметів за допомогою якихось засобів (вимірювальних приладів,

хімічних реактивів і т. д.) визначити окремі предмети зі специфічними властивостями. Така проблема може виникнути на митниці, контрольно-пропускних пунктах, у відділах технічного контролю, в соціальній сфері. На відміну від задач ОКР, такі задачі умовно можна назвати задачами необмеженого комбінаторного розпізнавання.

6.2. Обмежене комбінаторне розпізнавання

Розглянемо типову задачу обмеженого комбінаторного розпізнавання.

Задача 1. Нехай у наявності є n альтернатив прийняття економічних рішень $\mathbf{A} = \{A_1, A_2, \ldots, A_n\}$, про які відомо лише те, що серед них є m прийнятних ($m < n$). Існує механізм, який для фіксованого k ($k \leq m$) дозволяє визначити, чи існує серед альтернатив довільної k-вибірки хоча б одна неприйнятна. Необхідно за мінімальну кількість k-вибірок знайти k прийнятних альтернатив.

Наведемо декілька конкретних прикладів такої задачі.

Приклад 1. (Задача про лотерею.) Нехай задана множина натуральних чисел $N_n = \{1, 2, \ldots, n\}$... З неї випадково вибирається підмножина виграшних чисел $M = \{i_1, i_2, \ldots, i_m\}$ ($m < n$). Експеримент полягає у виборі k чисел ($k \leq m$) з N_n. Треба знайти мінімальну кількість таких k-вибірок, щоб хоча б одна з них належала M.

Приклад 2. (Задача про вимикачі.) Нехай є система, що складається з однієї лампочки та n вимикачів, незалежно одне від одного приєднаних до цієї лампочки. Відомо, що серед них m зіпсованих. Експеримент полягає в одночасному включенні k вимикачів ($1 < k \leq m$). Якщо серед них хоча б один справний, то лампочка засвічується. Необхідно за мінімальну кількість спроб знайти k несправних вимикачів.

Задачу про фальшиві монети також можна звести до задачі 1. Сформулюємо цю задачу в такій математичній постановці.

Нехай задана множина з n чисел $X=\{x_1, x_2, \ldots, x_n\}$, яка складається з m одиниць і $n - m$ нулів. Експеримент полягає у виборі ***фіксованої*** кількості k ($k \le m$) чисел. Ця фіксована кількість визначає добуток вибраних чисел, який дорівнює 0 або 1. Необхідно за мінімальну кількість спроб знайти k чисел, добуток яких дорівнює 1.

Стратегія оптимального розв'язування задач ОКР складається у такому розбитті вихідної множини чисел на групи, щоб потім, провівши експерименти за допомогою k-вибірок у кожній з них, знайти необхідну кількість визначених чисел [1].

Зупинимось на прикладі задачі про вимикачі. Позначимо мінімальну кількість спроб, за яку необхідно знайти k несправних вимикачів через $F_m^k(n)$.

Розглянемо приклад 2 для таких значень: $n = 9$, $m = 4$, $k = 2$.

Найпростішим розв'язком є такий. Пробуємо всі комбінації з двох вимикачів, поки не натрапимо на два зіпсованих, і тоді лампочка не засвітиться. Всього таких комбінацій $C_9^2 = 36$, серед них $C_4^2 = 6$ комбінацій із зіпсованими вимикачами. Отже, в найгіршому випадку через 31 спробу буде знайдено розв'язок задачі.

Більш вдалий розв'язок отримаємо, коли 9 вимикачів розіб'ємо, наприклад, на чотири групи (3 + 2 + 2 + 2). Спочатку зробимо $C_3^2+C_2^2+C_2^2+C_2^2 =$ 6 спроб. Якщо лампочка засвітиться кожний раз, то це може бути тільки тоді, коли в кожній групі буде по одному зіпсованому вимикачу. Беремо дві групи по 2 вимикачі і комбінуємо з них по 2 вимикачі, беручи по одному з кожної групи. Це потребує 4 спроби, а в сумі розв'язок отримаємо за 10 спроб.

Але існує ще один розв'язок, коли 9 вимикачів розбиваємо на три групи (3 + 3 + 3). Тоді обов'язково знайдеться група, в якій не менше двох зіпсованих вимикачів. Кількість спроб тепер становить $C_3^2+C_3^2+C_3^2 = 9$, що і буде оптимальним розв'язком. Іншими словами $F_4^2(9) = 9$.

Очевидно, що $F_m^m(n)=C_n^m$, тому що набір з одиниць єдиний і для його знаходження в найгіршому випадку потрібно перебрати всі комбінації.

Для $k = 2$ вже є досвід розв'язування задачі про вимикачі [1]. При цьому був знайдений спосіб (принцип оптимальності), за яким краще всього треба розбивати всю множину чисел на групи.

Принцип оптимальності: для $k = 2$ необхідно всю множину чисел розбити на стільки груп, щоб хоча в одній з них було не менш ніж дві одиниці.

Звідси витікає, що число груп повинно бути $m - 1$.

Позначимо $\lambda \equiv n \ (mod\ m - 1)$.

Лема 6.1. $$F_3^2(n) = \frac{n(n-2)+\lambda}{4}. \qquad (6.1)$$

Доведення. Нехай $\lambda \equiv n \equiv 0 \ (mod\ 2)$. Тоді множина з n чисел розбивається на дві однакові групи з $n/2$ чисел, і

$$F_3^2(n) = C_{n/2}^2 + C_{n/2}^2 = 2 \cdot \frac{n/2(n/2-1)}{2} = \frac{n(n-2)}{4}.$$

Якщо $\lambda \equiv 1 (mod\ 2)$, то множина з n чисел розбивається на дві різних групи з $\frac{n+1}{2}$ та $\frac{n-1}{2}$ числами відповідно.

Тоді

$$F_3^2(n) = C_{\frac{n+1}{2}}^2 + C_{\frac{n-1}{2}}^2 = \frac{1}{2}\left(\frac{n+1}{2}\cdot\frac{n-1}{2} + \frac{n-1}{2}\cdot\frac{n-3}{2}\right) = \left(\frac{n-1}{2}\right)^2.$$

Можна записати загальну формулу

$$F_3^2(n) = \frac{(n-\lambda)(n-2+\lambda)}{4} = \frac{n(n-2)+2\lambda-\lambda^2}{4}.$$

Враховуючи те, що $\lambda^2 \equiv \lambda (\mathrm{mod}\, 2)$, отримаємо формулу (6.1).

Лема 6.2. При поділі n на $m - 1$ груп отримаємо розбиття:

$$n = (m-1-\lambda)\left(\frac{n-\lambda}{m-1}\right) + \lambda\left(\frac{n-\lambda}{m-1}+1\right). \qquad (6.2)$$

Доведення. При діленні числа n на q отримаємо залишок $n (\mathrm{mod}\ q)$.

Отже $n = q\left\lfloor \frac{n}{q} \right\rfloor + n \pmod q$. Запишемо $q = [q - n \pmod q] + n \pmod q$, звідки $n = \left[q - n \pmod q\right]\left\lfloor \frac{n}{q} \right\rfloor + n \pmod q\left(\left\lfloor \frac{n}{q} \right\rfloor + 1\right)$. Підставляючи сюди $q = m - 1$ та $\left\lfloor \frac{n}{m-1} \right\rfloor = \frac{n-\lambda}{m-1}$, отримаємо шукану формулу (6.2).

Звідси легко вивести загальну формулу

$$n = \sum_{i=0}^{q-1} \left\lfloor \frac{n+i}{q} \right\rfloor. \tag{6.3}$$

Теорема 6.1. $$F_m^2(n) = \frac{(n-\lambda)(n+\lambda-m+1)}{2(m-1)}. \tag{6.4}$$

Доведення. Скористаємося результатами леми 6.2 при розбитті множини чисел на $m - 1$ груп.

$$F_m^2(n) = (m-1-\lambda)\, C^2_{\frac{n-\lambda}{m-1}} + \lambda C^2_{\frac{n-\lambda}{2}+1} = \frac{m-1-\lambda}{2}\left(\frac{n-\lambda}{m-1}\right)\left(\frac{n-\lambda}{m-1}-1\right) + \frac{\lambda}{2}\left(\frac{n-\lambda}{m-\lambda}+1\right)\left(\frac{n-\lambda}{m-1}\right).$$

Після скорочень отримаємо формулу (6.4).

Теорема 6.2. Нехай $m = 2r + 1$.

Тоді

$$F_{2r+1}^3(n) = \frac{1}{6r^2}(n-\lambda)(n-\lambda-r)(n-2r+2\lambda), \quad (r \ge 1). \tag{6.5}$$

Доведення. Для $k = 3$ отриманий розв'язок знаходитимемо, користуючись принципом оптимальності. Треба розбити n на r приблизно рівних груп, тоді хоча б в одній з них буде не менш ніж три одиниці. При цьому необхідно, щоб кожна група мала обсяг не менший ніж три. Тому з (6.3) випливає

$$F_{2r+1}^3(n) = \sum_{i=0}^{r-1} C^3_{\left\lfloor \frac{n+i}{r} \right\rfloor}. \tag{6.6}$$

Якщо скористатись параметром $\lambda \equiv n \pmod r$, то групи складатимуться з $r-\lambda$ чисел по $\frac{n-\lambda}{r}$ і λ чисел по $\frac{n-\lambda}{r}+1$.

Тому

$$F^3_{2r+1}(n)=(r-\lambda)\,C^3_{\frac{n-\lambda}{r}}+\lambda C^3_{\frac{n-\lambda}{r}+1}=\left(\frac{r-\lambda}{6}\right)\left(\frac{n-\lambda}{r}\right)\left(\frac{n-\lambda}{r}-1\right)\left(\frac{n-\lambda}{r}-2\right)+$$
$$+\frac{\lambda}{6}\left(\frac{n-\lambda}{r}+1\right)\left(\frac{n-\lambda}{r}\right)\left(\frac{n-\lambda}{r}-1\right). \qquad (6.7)$$

Спрощуючи цей вираз, отримаємо формулу (6.5).

Задача 2. Нехай задані дві множини чисел X = $\{x_1, x_2, \ldots, x_{n_1}\}$ та Y = $\{y_1, y_2, \ldots, y_{n_2}\}$, причому в першій множині міститься m_1 одиниць ($m_1 \le n_1$), а в другій – m_2 одиниць ($m_2 \le n_2$). Експеримент полягає у виборі k ($k \ge 1$) чисел з будь-яких множин ($k \le m_1 + m_2$), після чого стає відомим їх добуток. Необхідно за мінімальну кількість спроб знайти k чисел, добуток яких дорівнює 1.

Треба зазначити, що в загальному випадку цю задачу ще не розв'язано.

6.3. Необмежене комбінаторне розпізнавання

Наведемо формальну постановку задачі.

Нехай задана множина із n чисел X=$\{x_1, x_2, \ldots, x_n\}$, яка складається з m одиниць й $n - m$ нулів. Експеримент полягає у виборі довільної кількості k ($k \le m$) чисел, добуток яких дорівнює 0 чи 1. Необхідно за мінімальну кількість спроб (перевірок) знайти всі числа, які рівні 0 (або, що те саме — 1).

Наведемо приклад однієї такої практичної задачі.

Задача про радіоактивні предмети. Нехай через митницю проходить деякий вантаж з n предметів. Відомо, що k з них ($k < n$) радіоактивні. Для перевірки на радіоактивність можна вибирати довільну кількість предметів. У результаті експерименту одержуємо відповідь усього лише на питання, чи є в цій кількості предметів радіоактивний. Необхідно за мінімальне число перевірок знайти всі радіоактивні предмети.

Очевидно, що таку задачу легко розв'язати за n перевірок, вибираючи кожного разу по одному предмету. Але, так як кожна перевірка пов'язана з певними матеріальними витратами, то такий розв'язок буде найгіршим.

Розглянемо задачу про радіоактивні предмети для випадку $k=2$ на прикладі радіоактивних куль.

Нехай задано n куль, серед яких дві кулі радіоактивні (далі активні). Необхідно їх виявити за мінімальне число перевірок.

Сформулюємо загальні принципи, які застосовуються при розв'язуванні подібних задач [2].

1. Якщо з 2^s куль активна одна, то її можна знайти за s перевірок. На першому кроці перевіряється половина куль, потім методом дихотомії перевіряється та множина куль, де знаходиться активна куля.

2. Якщо куль більше, ніж 2^s, то за s кроків не можна забезпечити відшукання однієї активної кулі.

3. Якщо з n куль активні 2, то є $C_n^2=\dfrac{n(n-1)}{2}$ варіантів різних активних пар. Тому, якщо $\dfrac{n(n-1)}{2}>2^s$, то за s перевірок не вдасться знайти активну пару.

4. Якщо з n куль на першому кроці випробовуємо k куль, то це означає, що для отримання відповіді існує $C_{n-k}^{\ 2}$ варіантів для цих куль і $C_n^2-C_{n-k}^{\ 2}$ варіантів — для інших куль. Негативний результат перевірки k куль означає, що обидві активні кулі перебувають серед $n-k$, що залишилися. У будь-якому разі, якщо в розпорядженні залишилося тільки i випробувань, то обов'язково повинне бути $C_{n-k}^{\ 2}\le 2^i$ і $C_n^2-C_{n-k}^{\ 2}\le 2^i$.

5. Якщо за s перевірок удалося найти розв'язок для n куль, то для меншої кількості куль можна знайти розв'язок також за s перевірок. Це випливає з тих міркувань, що той самий результат можна досягти, якщо доповнити множину куль до числа n фіктивними (неактивними) кулями. Це означає, що для двох однакових значень s і різних значень n для всіх проміжних значень кількості куль задача розв'язується за ті самі s перевірок.

При розв'язуванні задачі будуватимемо стратегію послідовних випробувань. Якщо чергова перевірка куль ніяк (ні кількістю, ні індивідуальними властивостями) не залежить від попередньої перевірки, то назвемо таку стратегію незалежною. Якщо ж визначення кількості куль і їх індивідуальність залежить від результату попередньої перевірки, то назвемо таку стратегію покроковою.

Для незалежної стратегії визначальним є побудова розбивки n куль на m частин $R_n = (k_1, k_2, \ldots, k_m)$, де k_i — кількість куль, які випробовуються на i-му кроці $(1 \le i \le m)$, і $\sum_{i=1}^{m} k_i = n$. Після m випробувань задача зводиться або до отримання однієї множини (з меншою кількістю), що містить обидві активні кулі, або до двох множин, які містять по одній активній кулі.

Введемо позначення чотирьох функцій, які пригодяться у подальшому при розв'язуванні конкретних задач [3]:

$f_1(n)$ — мінімальне число перевірок для виявлення однієї активної кулі із n заданих;

$f_2(n)$ — мінімальне число перевірок для виявлення двох активних куль із n заданих;

$g(n_1, n_2)$ — мінімальне число перевірок для виявлення двох активних куль, які перебувають по одній у двох множинах, одна з яких містить n_1 куль, а друга — n_2;

$h(n_1, n_2)$ — мінімальне число перевірок для пошуку двох активних куль, якщо в першій множині є хоч б одна активна куля.

Легко підрахувати, що кількість варіантів для функції $h(n_1, n_2)$ $m = n_1 \cdot n_2 + C_{n_1}^2$.

Наведемо ряд тверджень, які необхідні для побудови оптимальних стратегій.

Для функцій $f_1(n)$ і $f_2(n)$ справедливі співвідношення:

$$f_1(2^k)=k,\ k\geq 1 \text{ і } f_2(2^k)=2k,\ k\geq 2.$$

Це легко перевіряється шляхом розбиття сукупності куль на дві рівні частини, з яких для подальшої перевірки залишається та частина, у якій виявлений позитивний результат перевірки. Частина куль з негативним результатом відкидається і в подальшому не перевіряється.

Для функції g справедливе співвідношення $g(n_1,\ n_2)\leq f_1(n_1)+f_1(n_2)$, причому можливо й строга нерівність, а також очевидним є співвідношення

$$g(2^k,2^s)=f_1(2^k)+f_1(2^s)=k+s,\ (k,s=1,2,...). \tag{6.8}$$

Якщо $g(n_1,\ n_2)=m$, то виконується рівність

$$g(2^k n_1,2^s n_2)=m+k+s\ \ (k,s=0,1,...). \tag{6.9}$$

Переходимо тепер до вивчення функції $f_2(n)$. Опишемо деякі результати оптимального пошуку двох активних куль для певних значень n, які відрізняються одне від одного стратегією покрокових дій. Функції $g(n_1,n_2)$ і $h(n_1,n_2)$ використовуються як допоміжні при дослідженні функції $f_2(n)$.

Необхідно зазначити, що для розв'язування задач пошуку двох активних куль використовуються як підходи на основі логічних міркувань, так і підходи, засновані на поняттях теорії графів [4].

З огляду на загальні принципи будь-яка активна пара може бути представлена ребром повного n-вершинного графа. Шляхом перевірок можна досягти зменшення кількості таких допустимих ребер. При відборі кількості куль для перевірки використовуються принципи, наведені в п. А, Б, В.

А. Якщо при перевірці кількості куль отриманий позитивний результат, то всі допустимі ребра починаються у цій групі, а інші ребра в подальшому не розглядаються (пропадають).

Б. Якщо отриманий негативний результат, то всі ребра, які починаються у цій групі, пропадають, а інші залишаються.

В. Число залишених чи не залишених допустимих ребер повинно бути в границі степеня двійки і на одиницю менше, ніж у попередній перевірці.

6.4. Розпізнавання двох радіоактивних куль

На основі цих підходів (логічного та графового) розроблені методи та алгоритми пошуку двох активних куль на множині заданих.

Наведемо деякі допоміжні результати у вигляді лем для функцій $g(n_1, n_2)$ і $h(n_1, n_2)$. Введемо позначення: $<n>$ означає, що перевіряється група з n куль, $<n_1>$ означає, що перевіряються n куль з 1-ї групи, $<m_2>$ — відповідно для 2-ї групи позначення $<n_1+m_2>$ означає, що перевіряється сумарна кількість куль з 1-ї та 2-ї груп відповідно.

Позначення $<\cdot>^+$ означає, що в групі взятих для перевірки куль виявлена хоча б одна активна, $<\cdot>^-$ означає, що в групі немає жодного активної кулі.

Лема 6.3. $g(3, 5)=4$.

Беремо для перевірки по одній кулі з кожної групи <1,1>. Якщо результат $<1,1>^-$, то активні кулі знаходимо серед інших за $f_1(2)+f_1(4)=3$ перевірки, в сумі отримаємо 4 перевірки.

Якщо результат $<1,1>^+$, то одну активну кулю знаходимо з цієї групи за одну перевірку. Для знаходження другої активної кулі перевіряємо чотири кулі, що залишилися у групі з п'яти куль. Якщо результат перевірки $<4>^+$, то п'ята куля із цієї групи неактивна, а активна одна з цієї четвірки. Її можна виявити за $f_1(4)=2$ перевірки, і в сумі буде 4 перевірки.

Якщо отримаємо результат $<4>^-$, то активна п'ята куля, а друга активна куля знаходиться із групи 3-х куль за $f_1(3)=2$ перевірки, що теж у сумі дає 4 перевірки.

Лема 6.4. $h(5, 4)=5$.

Перевіряємо кулі $<1_1+2_2>$, тобто одну кулю з 1-ї групи та дві — з 2-ї групи. Нехай результат перевірки буде $<1_1+2_2>^+$.

Беремо для перевірки кулю $<1_1>$. Якщо $<1_1>^+$, це означає, що одна куля вже визначена.

Перевіряємо кулі $<4_1>$. Якщо $<4_1>^+$, то друга активна куля визначається з цієї групи за $f_1(4_1)=2$ перевірки. У сумі отримаємо 5 перевірок.

Якщо $<4_1>^-$, тоді друга активна куля перебуває в другій групі, яку можна визначити за $f_1(4_2)=2$ перевірки. У сумі буде 5 перевірок.

Якщо на 2-му кроці результат перевірки буде $<1_1>^-$, то це означає, що $<2_2>^+$ та $<4_1>^+$ (за визначенням функції h). Дві активні кулі легко виявити за 3 перевірки: $f_1(4_1)=2$ і $f_1(2_2)=1$. У сумі буде 5 перевірок.

Якщо на 1-му кроці результат $<1_1+2_2>^-$, то поступаємо таким чином. Перевіряємо кулі $<2_2>$, що залишилися у 2-й групі. Якщо $<2_2>^+$, тоді 2 активні кулі легко виявити за 3 перевірки: $f_1(4_1)=2$ і $f_1(2_2)=1$. У сумі виходить 5 перевірок.

Якщо $<2_2>^-$, то дві активні кулі визначаються із 1-ї групи ($<4_1>^+$) за $f_2(4_1)=3$ перевірки У сумі також буде 5 перевірок.

Лема 6.5. $h(6, 2)=5$.

Беремо для перевірки дві кулі з 1-ї групи $<2_1>$. Якщо $<2_1>^+$, перевіряємо наступні 4 кулі з 6-ти. Якщо результат $<4_1>^+$, тоді дві активні кулі можна знайти за три перевірки: $f_1(4_1)=2$ і $f_1(2_1)=1$. У сумі отримуємо 5 перевірок.

При результаті $<4_1>^-$ беремо для перевірки дві кулі з 2-ї групи. Якщо перевірка дає результат $<2_2>^+$, тоді дві активні кулі легко виявити за 2 перевірки: $f_1(2_1)=1$ і $f_1(2_2)=1$. У сумі буде 5 перевірок.

Якщо результат $<2_2>^-$, то це означає, що 2 активні кулі знайдені (результат першої перевірки $<2_1>^+$).

Розглянемо функцію $f_2(n)$ для деяких конкретних значень. Позначимо результат перевірки знаком «+», якщо при перевірці виявлена хоча б одна

активна куля (позитивна перевірка), «–» — у протилежному випадку, в дужках — результати перевірок на попередніх кроках.

Для n = 3, 4, 5 легко знайти дві активні кулі, перевіряючи їх по одній. Отримаємо, що $f_2(3)=2$, $f_2(4)=3$, $f_2(5)=4$. Далі застосовуємо незалежну стратегію шляхом побудови розбивок і перевірки кожної групи окремо.

Для $n=6$ існує дві розбивки, що приводять до розв'язку: $R_1=(4,2)$ і $R_2=(2,2,2)$. Якщо в R_1 отримаємо результат$(+,-)$, то $f_2(6)=2+f_2(4)=5$, для результату $(+,+)$ отримаємо $f_2(6)=g(4,2)+2=5$.

Покажемо, що $f_2(7)=5$. До розв'язку приводить розбивка куль на 3 групи $R=(4,2,1)$. Перевіряємо 2 кулі. Якщо результат «–», то тоді $f_2(7)=1+f_2(5)=1+4=5$. Якщо результат 1-ї перевірки «+», то на другому кроці перевіряємо 4 кулі. При результат перевірки на другому кроці «–» перевіряємо останню в розбивці одну кулю. Якщо вона активна, то другу активну отримаємо за одну перевірку з групи 2-х, у протилежному разі активні обидві кулі з групи 2-х.

При результаті перевірки «+» приходимо до функції $g(4,2)=f_1(4)+f_1(2)=2+1=3$. А всього потрібно 5 перевірок.

Розглянемо функцію $f_2(n)$ для $n=9$ та $n=10$ і покажемо, що $f_2(9)=f_2(10)=6$.

$n=9$. У цьому випадку до успіху приводить розбивка $R=(4,4,1)$. Перевіряємо групи з 4-х куль. Якщо після цих перевірок буде результат $(+,+)$ то тоді $f_2(9)=2+g(4,4)=2+f_1(4)+f_1(4)=2+2+2=6$. Якщо результат перевірки цих груп буде $(+,-)$ чи $(-,+)$, то тоді беремо ту групу з 4-х куль, для якої результат перевірки «+», і одну останню неперевірену кулю. Дві активні кулі серед цих 5-ти виявляються за $f_2(5)=4$ перевірки, а разом буде $f_2(9)=2+f_2(5)=2+4=6$ перевірок.

$n=10$. До успіху приводить розбивка $R=(4,4,2)$. Так само, як у випадку $n=9$, перевіряємо групи з 4-х куль. Якщо після цих перевірок буде результат

$(+,+)$ то тоді $f_2(10)=2+g(4,4)=2+f_1(4)+f_1(4)=2+2+2=6$. Якщо результат перевірки цих груп буде $(+,-)$ чи $(-,+)$, то залишаємо кулі з результатом «+», і на третьому кроці перевіряємо групу з 2-х куль. При результаті «–» дві активні кулі з 4-х легко виявити за 3 перевірки. А якщо результат «+», то тоді $f_2(10)=3+g(4,2)=3+f_1(4)+f_1(2)=3+2+1=6$.

Отже маємо, що $f_2(10)=6$.

Для 11 куль 7 перевірок уже недостатньо. Легко можна показати, що для $11\le n\le 14$ $f_2(n)=7$. Розглянемо по черзі ці випадки.

$n=11$. Беремо для перевірки одну кулю. Якщо результат перевірки цієї кулі «–», то залишається 10 куль. І тоді $f_2(11)=1+f_2(10)=1+6=7$. При результаті перевірки «+» друга активна куля знаходиться серед 10-ти за 5 перевірок. Всього у цьому випадку достатньо 6 перевірок.

$n=12$. Беремо для перевірки дві кулі. Якщо результат перевірки цих куль «–», то, як і в першому випадку, залишається 10 куль і $f_2(12)=1+f_2(10)=1+6=7$. При результаті перевірки «+» на другому кроці беремо для перевірки 8 куль з 10-ти. Якщо результат перевірки цих куль «–», то дві активні кулі легко знаходяться серед 4-х за 3 перевірки. Якщо результат перевірки «+», то тоді 2 активні кулі легко знаходяться за $f_2(12)=2+g(8,2)=2+f_1(8)+f_1(2)=2+3+1=6$ перевірок.

$n=13$. Беремо для перевірки 3 кулі. Якщо результат перевірки цих куль «–», то, як і в першому випадку, залишається 10 куль і $f_2(13)=1+f_2(10)=1+6=7$. При результаті перевірки «+» на другому кроці беремо для перевірки 8 куль з 10-ти. Якщо результат перевірки цих куль «–», то дві активні кулі легко знаходяться серед 5-ти за 4 перевірки. Якщо результат перевірки 8-ми куль «+», то тоді 2 активні кулі легко знаходяться за $f_2(13)=2+f_1(8)+f_1(3)=2+3+2=7$ перевірок.

$n=14$. Беремо для перевірки 4 кулі. Якщо результат перевірки цих куль «–», то, як і в першому випадку, залишається 10 куль і $f_2(14)=1+f_2(10)=1+6=7$.

При результаті перевірки «+» на другому кроці беремо для перевірки 8 куль з 10-ти. Якщо результат перевірки цих куль «–», то дві активні кулі легко знаходяться серед 6-ти за 5 перевірок. Якщо результат перевірки 8-ми куль «+», то тоді 2 активні кулі легко знаходяться за $f_2(14) = 2 + f_1(8) + f_1(4) = 2 + 3 + 2 = 7$ перевірок.

$n = 15$. Для першої перевірки беремо 5 куль. Якщо результат перевірки «–», то з групи 10 куль, що залишилися, дві активні кулі можна знайти за $f_2(10) = 6$ перевірок, а разом буде 7 перевірок. При результаті перевірки «+» на другому кроці здійснюємо перевірку 6-ти куль з 10-ти. Якщо результат перевірки цих куль «+», то тоді приходимо до функції $g(n_1, n_2)$ зі значеннями $n_1 = 6$ і $n_2 = 5$. А так як $g(6, 5) = 5$ (див. лема 6.1 та форм. 9), то 2 активні кулі серед 15-ти можна знайти за $f_2(15) = 2 + g(6,5) = 2 + 5 = 7$ перевірок.

Якщо результат перевірки 6-ти куль негативний, то в цьому випадку приходимо до функції $h(n_1, n_2)$, у якій $n_1 = 5$ і $n_2 = 4$. За лемою 6.2 $h(5, 4) = 5$. Значить, що і в цьому випадку дві активні кулі можна знайти за 7 перевірок.

Покажемо, що для $17 \le n \le 22$ $f_2(n) = 8$.

Досить легко це показати для $17 \le n \le 19$. Розглянемо ці випадки.

$n = 16 + k$, $k = 1,2,3$. Беремо для перевірки $k+1$ кулю. Якщо результат перевірки цих куль негативний, то залишається 15 куль і дві активні кулі можна виявити за $f_2(16+k) = 1 + f_2(15) = 1 + 7 = 8$ перевірок. При результаті перевірки «+» 15 куль, що залишилися, розбиваємо на 4-ри групи (8, 4, 2, 1). Послідовно кожну групу перевіряємо доти, поки не буде знайдена група з активною кулею. Потім одну кулю знайдемо з цієї групи, а другу — з групи $(k+1)$ куль. Легко пересвідчитися, що на пошук 2-х активних куль знадобиться не більше ніж 8 перевірок.

$n = 20$. Беремо для перевірки 5 куль. Якщо результат перевірки цих куль «–», то залишається 15 куль і дві активні кулі можна виявити за $f_2(20) = 1 + f_2(15) = 1 + 7 = 8$ перевірок. При результаті перевірки «+» з 15-и куль, що залишилися, беремо для перевірки 12 куль. Якщо перевірка дає позитивний

результат, то тоді $g(12,5)=6$, і 2 активні кулі серед 20-ти можна знайти за $f_2(20)=2+g(12,5)=2+6=8$ перевірок. При результаті перевірки «–» з 15-ти куль залишається три. А це означає, що дві активні кулі серед 8-ми (5+3) можна визначити за $f_2(8)=6$ перевірок. Всього буде 8 перевірок.

$n=21$. Беремо для перевірки 6 куль. Якщо результат перевірки цих куль «–», то залишається 15 куль і дві активні кулі можна виявити за $f_2(21)=1+f_2(15)=1+7=8$ перевірок. При результаті перевірки «+» з 15-ти куль, що залишилися, беремо для перевірки 10 куль. Якщо перевірка дає позитивний результат, то тоді $g(10,6)=6$ (див. лема 6.1 та форм. 9), і 2 активні кулі серед 21-ї можна знайти за $f_2(21)=2+g(10,6)=2+6=8$ перевірок. Якщо на 2-му кроці результат перевірки «–», то з 15-ти залишається 5 куль. Перевіряємо ці кулі. Якщо результат перевірки «+», то дві активні кулі визначаються за $f_2(21)=3+g(6,5)=3+5=8$ перевірок. При негативному результаті залишається 6 куль, дві активних серед яких визначаються за 5 перевірок ($f_2(6)=5$). А всього для пошуку 2-х активних куль також знадобиться $f_2(21)=3+5=8$ перевірок.

Покажемо, що і серед 22-х куль 2 активні можна знайти за 8 перевірок, тобто, що $f_2(22)=8$.

Доведення цього твердження набагато складніше, ніж у попередніх випадках, і проводиться таким чином. На першому кроці беремо для перевірки 7 куль. Якщо результат перевірки буде «–», то дві активні кулі знаходяться з групи 15-ти за $f_2(15) = 7$ перевірок. А в сумі отримаємо 8 перевірок, що і потрібно було довести.

Якщо результат перевірки «+», то треба довести, що $h(7, 15) = 7$. У цій ситуації до мети приводить тільки один варіант другої перевірки. Беремо 3 кулі з 7-ми та одну кулю з 15-ти.

Якщо друга перевірка дає результат «–», то порівняно просто можна показати, що $h(4, 14) = 6$. Справді, тепер для перевірки на 3-му кроці беремо 8 куль із 14-ти. Якщо результат перевірки буде «+», то очевидно, що $g(4, 8) = 5$, якщо результат 3-ї перевірки буде «–», то потрібно довести, що $h(4, 6) = 5$.

Для четвертої перевірки беремо 4 кулі з 6-ти. Якщо результат «+», то очевидно, що $g(4, 4) = 4$. Якщо результат «–», то залишається показати, що $h(4, 2) = 4$. А це так і є, тому що для п'ятої перевірки залишається взяти останні 2 кулі. Якщо результат «+», то $g(4, 2) = 3$, а якщо «–», то $f_2(4) = 3$.

Тепер повернемося до випадку, коли результат перевірки на другому кроці позитивний.

Для подальших міркувань перенумеруємо кулі числами: 1, 2, ..., 22. Нехай перша перевірка куль < 16, 17, 18, 19, 20, 21, 22 > дає позитивний результат. Точно також дає позитивний результат друга перевірка куль < 15, 20, 21, 22 >.

Тепер для третьої перевірки беремо кулі < 1, 2, 3, 4, 5, 6, 7, 8, 16, 17 >.

У випадку позитивного результату для четвертої перевірки беремо кулі < 1, 2, 3, 4, 16 >. У випадку ж негативного результату третьої перевірки для четвертої перевірки беремо кулі < 9, 10, 11, 12, 18 >.

Залежно від результатів третьої й четвертої перевірок на 5-му кроці можливий один із чотирьох варіантів, для яких у дужках знаками +, – вказані результати перевірок на попередніх кроках. Розглянемо ці варіанти.

1. (+, +, +, +). Беремо для перевірки кулі < 16, 20 >. Якщо результат «–», то одна активна знаходиться серед куль з номерами 1, 2, 3, 4, а друга — 21 або 22, які можна виявити за 3 перевірки.

Якщо результат п'ятої перевірки «+», то для шостої перевірки береться куля 16. Якщо результат «+», то ця куля активна, а другу активну знайдемо серед куль < 15, 20, 21, 22 > за 2 перевірки. Якщо шоста перевірка «–», то одна активна, це куля з номером 20, а друга знаходиться серед куль < 1, 2, 3, 4 > за 2 перевірки.

2. (+, +, +, –). Беремо для перевірки кулі < 17, 20 >. Якщо результат «–», то одна активна знаходиться серед куль з номерами 5, 6, 7, 8, а друга — 21 або 22, які можна виявити за 3 перевірки.

Якщо результат п'ятої перевірки «+», то для шостої перевірки береться куля 17. Якщо результат «+», то ця куля активна, а другу активну знайдемо серед куль < 15, 20, 21, 22 > за 2 перевірки. Якщо шоста перевірка «–», то одна

активна, це куля з номером 20, а друга знаходиться серед куль < 5, 6, 7, 8 > за 2 перевірки.

3. (+, +, –, +). Беремо для перевірки кулі 18, 20. Якщо результат «–», то одна активна знаходиться серед куль з номерами < 9, 10, 11, 12 >, а друга активна — серед куль <21, 22>, які можна виявити за 3 перевірки.

Якщо результат п'ятої перевірки «+», то для шостої перевірки береться куля 18. Якщо результат «+», то ця куля активна, а другу активну знайдемо серед куль < 15, 20, 21, 22 > за 2 перевірки. Якщо шоста перевірка «–», то одна активна, це куля з номером 20, а друга знаходиться серед куль < 9, 10, 11, 12 > за 2 перевірки.

У кожному із цих трьох випадків задача розв'язується або знаходженням однієї кулі із двох на шостій перевірці, або однієї кулі із чотирьох на сьомій і восьмій перевірках, за умови негативного результату п'ятої перевірки. Якщо ж п'ята перевірка дає позитивний результат, то шоста перевірка визначає одну активну кулю, а сьома та восьма йдуть на пошук другої кулі серед чотирьох. Залишається розглянути випадок, коли третя та четверта перевірки дають негативний результат (+, +, –, –). Це означає, що в групі куль < 15, 20, 21, 22 > є хоча б одна активна (позитивний результат 2-го кроку) та (з урахуванням негативних результатів на 3-му та 4-му кроках) є група куль < 13, 14, 19 >, про яку нічого невідомо.

4. (+, +, –, –). Сформовану ситуацію зручно показати за допомогою графа (рис. 6.1). Вершини графа — номери куль. Ребра графа — можливі пари активних куль.

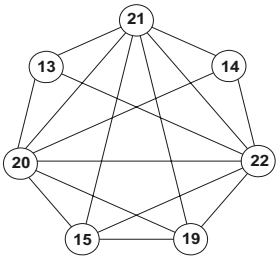


Рис. 6.1

Далі поступаємо так. З аналізу графа видно, що для п'ятої перевірки є шість варіантів вибору куль: < 13, 20 >, < 13, 21 >, < 13, 22 >, < 14, 20 >, < 14, 21 >, < 14, 22 >.

Починати можна з будь-якого варіанту, це неважливо. Нехай, наприклад, п'ята перевірка — кулі <13, 20>.

Як при результаті «–», так і при «+» для подальшого розгляду залишається підграф, що містить по 8 ребер (рис. 6.2). Тепер нескладно спланувати останні три перевірки. Так при отриманні результатів п’ятої перевірки «–» вибираємо для шостої перевірки кулю 22. Якщо результат перевірки «+», то ця куля активна, а другу активну кулю знаходимо серед куль < 14, 15, 19, 21 > за два кроки. Якщо результат перевірки кулі 22 «–», то тоді хоча б одна активна є серед куль < 15, 21 >.

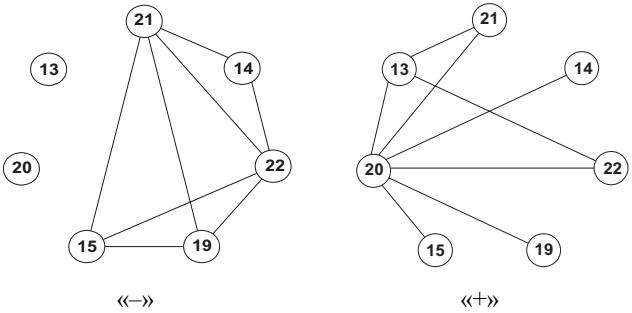


Рис. 6.2

Для сьомої перевірки беремо кулю 19. Якщо результат «+», то ця куля активна, а другу активну визначаємо серед куль < 15, 21 > за одну перевірку. Якщо результат «–», тоді куля 21 активна, а друга визначається за одну перевірку серед куль < 14, 15 >.

При отриманні при отриманні результаті п’ятої перевірки «+» для шостої вибираємо кулі < 21, 22 >. Якщо результат перевірки цих куль «+», тоді одна активна куля визначається серед куль <13, 20>, друга – серед куль < 21, 22 >. Разом буде 8 перевірок.

Якщо результат перевірки куль < 21, 22 > «–», тоді куля 20-та активна, а друга активна визначається з групи куль < 13, 14, 15, 19 > за дві перевірки. Всього перевірок також 8.

Розглянуто всі можливі варіанти, і тим самим доведено, що $f_2(22) = 8$.

Результати цих та інших досліджень показано в таблиці 6.1.

Табл. 6.1

N	3	4	5	6–7	8–10	11–15	16–22	23–31	32–44	45–63	64–89	90–127
$f_2(n)$	2	3	4	5	6	7	8	9	10	11	12	13

6.5. Деякі результати пошуку трьох та чотирьох активних куль

Задача пошуку трьох та чотирьох активних куль значно складніша, ніж задача пошуку двох активних куль. За аналогією вводяться такі позначення:

$f_3(n)$, $f_4(n)$ — мінімальна кількість спроб для виявлення відповідно трьох і чотирьох активних куль серед n заданих;

$h_3(k^+, n-k)$, $h_4(k^+, n-k)$ — мінімальна кількість спроб для виявлення відповідно трьох і чотирьох активних куль серед n заданих після того, як перевірка k куль дала позитивний результат.

Якщо з n куль активні 3, то є $C_n^3 = \dfrac{n(n-1)(n-2)}{6}$ варіантів різних активних трійок. Тому, якщо $\dfrac{n(n-1)(n-2)}{6} > 2^s$, то за s випробувань не вдасться знайти активну трійку.

Якщо з n куль на першому кроці випробовуємо k куль, то результат випробування «–» відповідає C_{n-k}^3 варіантів (три активні кулі перебувають серед $n-k$, що залишилися), а результат «+» — іншим $C_n^3 - C_{n-k}^3$ варіантам.

Для обчислення функцій $f_3(n)$ та $f_4(n)$ використовуються результати обчислень функції $f_2(n)$, а кращою стратегією є індуктивний метод, коли

загальна задача з даною кількістю активних куль зводиться до задачі з меншою кількістю активних куль. Очевидними є такі співвідношення:

$$h_3(2^+,k)=1+f_2(k+1),\ h_4(2^+,k)=1+f_3(k+1).$$

Наведемо результати досліджень для функції $f_3(n)$. Для $4\le n\le 9$ безпосередньо переконуємося, що $f_3(n)=n-1$.

Для значень $10\le n\le 25$ доведені та отримані такі результати:

$f_3(10)=8,\ f_3(11)=f_3(12)=f_3(13)=9,\ f_3(14)=f_3(15)=f_3(16)=10,$

$f_3(17)=f_3(18)=f_3(19)=f_3(20)=11, f_3(21)=f_3(22)=f_3(23)=f_3(24)=f_3(25)=12.$

А тепер щодо функції $f_4(n)$. Тут задача виявлення 4-х активних куль ще складніша. І поки що отримані результати для наступних значень n.

Безпосередньо переконуємося, що $f_4(n)=n-1$ для $5\le n\le 11$.

Для значень $13\le n\le 18$ доведені та отримані такі результати:

$f_4(12)=f_4(13)=11,\ f_4(14)=f_4(15)=12,\ f_4(16)=f_4(17)=f_4(18)=13.$

6.6. Висновки

Розглянуті задачі обмеженого та необмеженого комбінаторного розпізнавання предметів з нестандартними властивостями серед маси однотипних.

Приводиться формальна постановка задачі пошуку двох предметів з нестандартними властивостями на прикладі радіоактивних куль. Для розв'язування задачі використовуються як підходи на основі логічних міркувань, так і підходи, засновані на поняттях теорії графів. Описані алгоритми оптимального пошуку двох радіоактивних куль на множині заданих для $n\le 22$, отримані деякі результати пошуку трьох і чотирьох радіоактивних куль.

Результати досліджень можуть застосовуватися у різних сферах, пов'язаних з виявленням нестандартних предметів серед маси однотипних, зокрема на митниці, контрольно-пропускних пунктах, у відділах технічного контролю, соціальній сфері.

Список літератури

1. Донец Г. А. Задачи комбинаторного распознавания. Материалы XVI Международной конф. Проблемы теоретической кибернетики. Нижний Новгород, 2011. С. 142–144.
2. Билецкий В. И., Донец Г. А., Ненахов Э. И. Об одной задаче неограниченного комбинаторного распознавания. Теорія оптимальних рішень. 2013. С. 88–94.
3. Донец Г. А., Билецкий В. И., Ненахов Э. И. Оптимальный поиск двух активных шаров на множестве заданных. Теорія оптимальних рішень. 2015. С. 134–139.
4. Донец Г. А. Основы теории графов. Кировоград: ЧП «Эксклюзив-Систем», 2013. 280 с.

Розділ 3. КВАДРАТИЧНІ ТА ІГРОВІ ЗАДАЧІ

7. ПРО ТОЧНІ ДВОЇСТІ ОЦІНКИ ДЛЯ КВАДРАТИЧНИХ ЕКСТРЕМАЛЬНИХ ЗАДАЧ

О. А. Березовський

Анотація. Робота присвячена дослідженню двоїстих оцінок для квадратичних екстремальних задач загального вигляду. Сформульовано умови, при виконанні яких значення глобального екстремуму квадратичної екстремальної задачі та її двоїстої оцінки збігаються. Наведено приклади їх застосування до конкретних задач для визначення випадків, коли знаходження двоїстої оцінки дозволяє знайти розв'язок задачі.

Annotation. The paper is devoted to the study of dual estimates for quadratic extremal problems of general form. Conditions are formulated, where the values of the global extremum of quadratic extremal problem and its dual estimates coincide. Examples of their application to specific problems are given for determining the cases, when finding dual estimates makes possible to find solution for the problem.

7.1. Вступ

Під квадратичною екстремальною задачею розуміють задачу математичного програмування, у якої цільова функція та всі функції обмежень квадратичні:

$$f^* = \inf_{x \in T} f_0(x), \tag{7.1}$$

де $T = \{x : f_i(x) \le 0, i \in I^{LQ}, f_i(x) = 0, i \in I^{EQ}; x \in R^n\}$, $f_i(x) = x^T A_i x + b^T{}_i x + c_i$, $i \in \{0\} \bigcup I^{LQ} \bigcup I^{EQ}$ — квадратичні функції із симетричними $n \times n$-матрицями A_i, векторами $b_i \in R^n$ і константами $c_i \in R^1$; $m = |I^{LQ}| + |I^{EQ}|$ — загальна кількість обмежень. У загальному випадку квадратична екстремальна задача відноситься

до класу NP-складних задач. В останні десятиріччя значна увага зосереджена на дослідженні цього класу задач за допомогою різного типу опуклих релаксацій, зокрема SDP-релаксацій (semidefinite programming relaxation problems), SOCP-релаксацій (second-order cone programming relaxation problems), лагранжевих релаксацій (lagrangian relaxation problems) та інше. Одним із основних напрямків цих досліджень є виділення спеціальних підкласів квадратичних задач, для яких релаксації дозволяють знайти значення глобального екстремуму. В цій роботі наведені результати в цьому напрямку, пов'язані з двоїстою оцінкою ψ^* [1, 2] для квадратичної задачі (7.1) (інколи її називають двоїстою оцінкою Шора), яка визначається таким чином:

$$\psi^* = \sup_{u \in \bar{D} \cap U^+} \psi(u) \le \inf_{x \in T} f_0(x) = f^*, \tag{7.2}$$

де

$$\psi(u) = \inf_x L(u, x); \tag{7.3}$$

$L(u,x) = x^T A(u)x + b^T(u)x + c(u)$ — функція Лагранжа для задачі (7.1), в якій

$$A(u) = A_0 + \sum_{i=1}^{m} u_i A_i,\ b(u) = b_0 + \sum_{i=1}^{m} u_i b_i,\ c(u) = c_0 + \sum_{i=1}^{m} u_i c_i;$$

U^+ — область визначення вектора множників Лагранжа $u \in R^m$, яка враховує наявність обмежень у вигляді нерівностей: $U^+ = \{u : u_i \ge 0, i \in I^{LQ}\}$; $D = \{u : A(u) \succ 0\}$ ($\bar{D} = \{u : A(u) \succ= 0\}$) — множина змінних $u \in R^m$, при яких матриця $A(u)$ додатно (невід'ємно) визначена. Задачу (7.2) також називають лагранжевою релаксацією [3] (якщо бути точним, вона є лагранжевою релаксацією задачі (7.1) за всіма обмеженнями з виписаною у явному вигляді умовою, яка з точністю до граничних точок задає множину двоїстих змінних, при яких розв'язок внутрішньої задачі (7.3) обмежений знизу).

Якщо для опуклих задач двоїстий підхід дозволяє отримати як значення, так і точку глобального екстремуму, то в неопуклому випадку питання якості (точності) отриманої оцінки достатньо складне. В роботі наведені умови, при

виконанні яких значення глобального екстремуму квадратичної екстремальної задачі та її двоїстої оцінки збігаються. Наведено також приклади їх застосування до конкретних задач для визначення випадків, коли знаходження двоїстої оцінки дозволяє знайти їх розв'язок.

7.2. Необхідна та достатня умова точної двоїстої оцінки для квадратичної екстремальної задачі

У роботі [4] сформульовано та доведено умову отримання точної двоїстої оцінки для довільної квадратичної екстремальної задачі (7.1) (тобто, коли розрив двоїстості дорівнює нулю – $\psi^* = f^*$).

Теорема 7.1 [4, теорема 4]. Для того, щоб квадратична екстремальна задача з $f^* > -\infty$ мала точну двоїсту оцінку, необхідно і достатньо, щоб існував такий вектор множників Лагранжа u^*, при якому функцію $L(u^*, x) - f^*$ можна представити у вигляді суми квадратів лінійних функцій:

$$\exists u^* : L(u^*, x) - f^* = \sum_{i=1}^{k} l_i^2(x). \qquad (7.4)$$

Доведення. *Необхідність.* Нехай для квадратичної екстремальної задачі (7.1) $\psi^* = f^*$. При $u \to u^*$, де $u^* = \arg \sup_{u \in \overline{D} \cap U^+} \psi(u)$, розв'язок $x(u)$ внутрішньої задачі (7.3), який знаходиться шляхом розв'язання системи лінійних рівнянь $L_x(u, x) = 2A(u)x + b(u) = 0$ (відзначимо, що при $u \in D \cap U^+$ матриця $A(u)$ невироджена і система має єдиний розв'язок), прямує до деякої точки x^* з області, яка задається системою $2A(u^*)x + b(u^*) = 0$. Причому x^* не обов'язково буде точкою мінімуму початкової задачі і навіть може бути недопустимою точкою для задачі (7.1). Так як

$$L(u, x) = x^T A(u)x + b(u)^T x + c(u) =$$

$$= (x - x(u))^T A(u)(x - x(u)) + c(u) - x(u)^T A(u)x(u),$$

при u^* маємо

$$L(u^*,x) = (x-x^*)^T A(u^*)(x-x^*) + c(u^*) - x^{*T} A(u^*)x^* = (x-x^*)^T A(u^*)(x-x^*) + \psi^* =$$
$$= (x-x^*)^T A(u^*)(x-x^*) + f^* = \sum_{i=1}^{n} \lambda_i^* (\xi_i^*, x-x^*)^2 + f^*,$$

де λ_i^* — власні числа , а ξ_i^* — власні вектори матриці $A(u^*)$. (Відзначимо, що у випадку виродженості матриці $A(u^*)$ лінійні члени відсутні, оскільки в протилежному випадку $\psi(u^*) = -\infty$.) Так як усі власні числа матриці $A(u^*)$ невід'ємні ($u^* \in \overline{D}$), шукане розкладання (7.4) отримано.

Достатність. Нехай існує таке $\tilde{u} \in U^+$, що $L(\tilde{u},x) - f^* = \sum_{i=1}^{k} l_i^2(x)$, де $l_i(x)$ — лінійні функції. Тоді

$$\psi(\tilde{u}) = \min_x L(\tilde{u},x) = \min_x \sum_{i=1}^{k} l_i^2(x) + f^* \geq f^*.$$

Але за визначенням (7.3) функція $\psi(u)$ є оцінкою знизу для f^* при всіх $u \in \overline{D} \cap U^+$, тобто $\psi(\tilde{u}) = f^*$. Причому $\tilde{u} = u^*$ і $l_i(x^*) = 0$, $i = \overline{1,k}$.

Доведення достатності умови (7.4) і теореми 7.1 загалом завершено.

З доведення необхідності в теоремі 1 очевидно випливає таке твердження.

Наслідок 7.1. Якщо оцінка точна, то $L(u^*,x) - f^*$ можна представити у вигляді

$$L(u^*,x) = \sum_{i=1}^{n} \lambda_i(u^*)(\xi_i(u^*), x - x(u^*))^2 + f^*$$

де $\lambda_i(u^*)$, $i = \overline{1,n}$, — власні числа матриці $A(u^*)$, $\xi_i(u^*)$, $i = \overline{1,n}$, — власні вектори матриці $A(u^*)$, $x(u^*)$ — точка, до якої прямує послідовність розв'язків $x(u) = -A^{-1}(u)b(u)/2$ внутрішньої задачі (7.3) при $u \to u^*$. Причому $x(u^*)$ не обов'язково задовольняє обмеженням початкової задачі і, відповідно, не обов'язково є її розв'язком.

Слід зазначити, що цей варіант розкладання $L(u^*,x) - f^*$ на суму квадратів лінійних форм може бути не єдиним.

Наведемо приклади практичного застосування теореми 7.1.

Приклад 1. Глобальний мінімум полінома. В роботі [1] для знаходження глобального мінімуму обмеженого знизу полінома $P_0(x)$ ($x \in R^n$, s — вектор старших степенів полінома $P_0(x)$) була запропонована така схема побудови еквівалентної квадратичної задачі:

1) для усіх $\alpha^{(i)} \le \overline{\alpha} = s/2$ вводяться змінні

$$R(\alpha^{(i)}) = R(\alpha^{(j)})R(\alpha^{(k)}),$$

$$\alpha^{(i)} = \alpha^{(j)} + \alpha^{(k)},$$

де невід'ємний цілочисельний вектор $\alpha^{(r)} = (\alpha_1^{(r)}, \alpha_2^{(r)}, \ldots, \alpha_n^{(r)})^T \le \overline{\alpha}$ визначає змінну $R(\alpha^{(r)})$, якій у початковому просторі R^n відповідає моном $R(\alpha^{(r)}) = x_1^{\alpha_1^{(r)}} x_2^{\alpha_2^{(r)}} \ldots x_n^{\alpha_n^{(r)}}$. У результаті цього отримуємо повний набір змінних, які покривають усі мономи степені $\alpha^{(i)} \le \overline{\alpha}$, і поліном $P_0(x)$ можна представити у вигляді квадратичної функції $f_0(R)$ (відзначимо, що всі компоненти вектора s парні — у протилежному випадку поліном буде не обмежений знизу);

2) до квадратичних обмежень, які визначають нові «мономні» змінні, додається повне сімейство обмежень

$$R(\alpha^{(i)})R(\alpha^{(j)}) - R(\alpha^{(k)})R(\alpha^{(l)}) = 0$$

$$\forall \alpha^{(i)}, \alpha^{(j)}, \alpha^{(k)}, \alpha^{(l)}, \text{ таких що } \alpha^{(i)} + \alpha^{(j)} = \alpha^{(k)} + \alpha^{(l)},$$

тобто за допомогою їх лінійної комбінації можна отримати всі можливі представлення (степені меншої або рівної двом) будь-якого монома $x_1^{\alpha_1^{(r)}} x_2^{\alpha_2^{(r)}} \ldots x_n^{\alpha_n^{(r)}}$ степені $\alpha^{(r)} \le s$, а відповідно і полінома $P_0(x)$, у нових змінних.

У результаті задача $f^* = \min\limits_{x \in R^n} P_0(x)$ зводиться до квадратичної екстремальної задачі

$$f^* = \min_R f_0(R) \tag{7.5}$$

при обмеженнях

$$R(\alpha^{(i)})R(\alpha^{(j)}) - R(\alpha^{(k)})R(\alpha^{(l)}) = 0, \tag{7.6}$$

$$\alpha^{(i)} + \alpha^{(j)} = \alpha^{(k)} + \alpha^{(l)},$$

$$0 \le \alpha^{(r)} = (\alpha_1^{(r)}, \alpha_2^{(r)}, \ldots, \alpha_n^{(r)})^T \le s/2.$$

Відзначимо, що обмеження (7.6) включають обмеження як з пункту 2), так і з пункту 1), оскільки $R(0) = 1$.

У роботі [1] отримано такий результат.

Теорема 7.2 [1, с. 141]. Для того, щоб обмежений знизу поліном $P_0(x)$ володів Ω-властивістю, необхідно і достатньо, щоб поліном $P_0(x) - f^*$ можна було представити у вигляді суми квадратів поліномів.

Зауваження. У формулюванні теореми 7.2 використовується таке визначення: казатимемо, що поліном $P_0(x)$ володіє Ω-властивістю, якщо двоїста оцінка квадратичної задачі (7.5)–(7.6), яка відповідає задачі $f^* = \min\limits_{x \in R^n} P_0(x)$, є точною [1, с. 130].

Покажемо, як за допомогою теореми 7.1 довести теорему 7.2. В цьому випадку доведення значно спрощується — на відміну від доведення у [1] воно не потребує доведення допоміжної теореми про зміщення і розгляд випадку, коли мінімумом полінома є нульова точка. Дійсно, необхідна і достатня умова точності двоїстої оцінки квадратичної задачі (7.5)–(7.6) відповідно до теореми 7.1 формулюється так: $\exists u^* : L(R, u^*) - f^* = \sum\limits_{i=1}^{k} l_i^2(R)$. Підставимо замість змінних $R(\alpha)$ відповідні мономи в початкових змінних x. При цьому лінійні функції $l_i(R)$ приймуть вигляд поліномів — $l_i(R) = P_i(x)$, а в лівій частині рівності всі доданки з двоїстими змінними стануть рівними нулю, оскільки в усіх обмеженнях $R(\alpha^{(i)})R(\alpha^{(j)}) - R(\alpha^{(k)})R(\alpha^{(l)}) = 0$, $\alpha^{(i)} + \alpha^{(j)} = \alpha^{(k)} + \alpha^{(l)}$, обидва члени є виразами одного і того самого монома, вираженого в різних змінних, тобто $L(R, u) = P_0(x)$. Таким чином, з умови (7.4) для задачі (7.5)–(7.6) випливає

$P_0(x) - f^* = \sum_{i=1}^{k} P_i^2(x)$ і, відповідно, необхідність умови точності двоїстої оцінки квадратичної задачі (7.5)–(7.6), сформульованої в теоремі 7.2, доведена.

Тепер розглянемо достатність умови теореми 7.2. Підстановки правильні і в зворотному напрямку, тобто, якщо б розкладання $P_0(x) - f^* = \sum_{i=1}^{k} P_i^2(x)$ було відомим, то достатньо при побудові квадратичної задачі (7.5)–(7.6) замінити мономи поліномів $P_i(x)$, $i = \overline{1,k}$, на відповідні змінні $R(\alpha)$, щоб отримати квадратичну задачу з цільовою функцією $\tilde{f}_0(R)$, для якої справедливий вираз (7.4) при $u^* = 0$. Оскільки спочатку таке розкладання невідоме (а воно за припущенням існує), саме з метою його знаходження (точніше, знаходження $\tilde{f}_0(R)$) і вводяться обмеження вигляду $R(\alpha^{(i)})R(\alpha^{(j)}) - R(\alpha^{(k)})R(\alpha^{(l)}) = 0$, $\alpha^{(i)} + \alpha^{(j)} = \alpha^{(k)} + \alpha^{(l)}$ — за допомогою лінійної комбінації цих обмежень у функції Лагранжа мономи на початку довільно представленої цільової функції $f_0(R)$ ($P_0(x)$ у змінних R визначається неоднозначно) приводяться до необхідного для виділення квадратів вигляду, або, іншими словами, функція $f_0(R)$ приводиться до вигляду $\tilde{f}_0(R)$.

Таким чином, необхідна і достатня умова точності двоїстої оцінки з теореми 7.1 для квадратичної задачі спеціального вигляду (7.5)–(7.6) еквівалентна умові $P_0(x) - f^* = \sum_{i=1}^{k} P_i^2(x)$, що і треба було довести.

Приклад 2. Лінійна задача комплементарності. Лінійна задача комплементарності полягає у знаходженні вектора $x \in R^n$, який задовольняє умови

$$Mx + q \geq 0,\ x \geq 0,\ x^T(Mx + q) = 0,$$

для заданих матриці M і вектора q.

Беручи до уваги неоднозначність формулювання цієї задачі у вигляді квадратичних екстремальних задач, можна отримувати різні результати під час знаходження їх двоїстих оцінок. Наприклад, для широковідомої задачі-аналога

$$f^* = \min\left(x^T \left(\frac{M + M^T}{2} \right) x + x^T q) \right),$$

при обмеженнях

$$Mx + q \geq 0,$$

$$x \geq 0,$$

якщо матриця $\left(\frac{M + M^T}{2} \right)$ додатно визначена, в результаті розв'язання задачі (7.2)–(7.3) отримуємо як точну двоїсту оцінку $\psi^* = f^* = 0$, так і точку мінімуму, яка є розв'язком лінійної задачі компліментарності. Якщо ж ця матриця не є невід'ємно визначеною, двоїста оцінка не визначена.

Розглянемо іншу еквівалентну квадратичну постановку:

$$f^* = \min_{x \in R^n, y \in R^n} (y - Mx - q)^T (y - Mx - q),$$

при обмеженнях

$$x \geq 0,\ y \geq 0,$$

$$x_i y_i = 0,\ i = \overline{1,n}.$$

Якщо для цієї задачі $f^* = 0$, то множина її розв'язків збігається з множиною розв'язків вихідної лінійної задачі комплементарності. У цьому випадку двоїста оцінка є точною згідно з теоремою 7.1, наприклад при $u^* = 0$. Однак при такому значенні u^* ми не отримуємо шукані x^* і y^* (так як оптимальне значення двоїстої оцінки $\psi^* = 0$ досягається при довільному $\tilde{x}$ і $\tilde{y} = M\tilde{x} + q$.), до знаходження яких можливо приведе відповідне збурення цільової функції задачі. Якщо $f^* > 0$ (відповідає випадку, коли лінійна задача комплементарності не має розв'язків), то ψ^* може приймати як додатне

значення, що дозволяє однозначно константувати факт відсутності розв'язку початкової задачі, так і, на жаль, дорівнювати нулю, що не дає можливості щонебудь стверджувати.

Відзначимо, що наведені міркування узагальнюються на випадок задачі знаходження розв'язку довільної системи поліноміальних рівнянь та нерівностей $\begin{cases} P_i(x) = 0, & i \in I^E \\ P_i(x) \le 0, & i \in I^L \end{cases}$, якщо поставити їй у відповідність задачу мінімізації квадратів нев'язок рівностей та/або квадратів вигляду $(y_i - P_i(x))^2$ (y_i — допоміжні змінні) з відповідними обмеженнями та представити всі функції у квадратичному вигляді за допомогою схеми з прикладу 1 (використовуючи додаткові змінні $R(\alpha)$). Незважаючи на неоднозначність побудови таким чином квадратичної задачі-аналогу, для всіх постановок з такою цільовою функцією, якщо система має розв'язок (тобто значення естремуму дорівнює нулю), двоїста оцінка точна. У протилежному випадку оцінка може бути як більше від нуля (розв'язку системи не існує), так і дорівнювати нулю (тоді висновок неоднозначний).

7.3. Необхідна та достатня умова точної двоїстої оцінки для квадратичної екстремальної задачі в матричному вигляді

У деяких випадках може бути кориснішим використовувати формулювання необхідної і достатньої умови отримання точної двоїстої оцінки ψ^* для квадратичної задачі (7.1) (теорема 7.1) у матричному вигляді.

Теорема 7.3 [5]. Для того, щоб двоїста оцінка ψ^* (7.2)–(7.3) для квадратичної задачі (7.1) була точною, необхідно і достатньо, щоб матрицю $A_v(v^*) = \begin{pmatrix} A_0 & b_0/2 \\ b_0^T/2 & -f^* \end{pmatrix}$ можна було представити у вигляді різниці невід'ємно

визначеної матриці $A_{\bar{D}}$ і лінійної комбінації матриць $\overline{A}_i = \begin{pmatrix} A_i & b_i/2 \\ b_i^T/2 & c_i \end{pmatrix}$, $i = \overline{1,m}$, коефіцієнтами якої будуть координати вектора двоїстих змінних $u^* \in U^+$: $A_v(v^*) = A_{\bar{D}} - \sum_{i=1}^{m} u_i^* \overline{A}_i$.

Доведення. Трансформуємо задачу (7.1) в еквівалентну задачу однорідного вигляду:

$$f^* = \min_{(x,y)\in R^{n+1}} (x^T A_0 x + b_0^T xy), \tag{7.7}$$

при обмеженнях

$$x^T A_i x + b_i^T xy + c_i y^2 \le 0,\ i \in I^{LQ}, \tag{7.8}$$

$$x^T A_i x + b_i^T xy + c_i y^2 = 0,\ i \in I^{EQ} \tag{7.9}$$

$$y^2 - 1 = 0, \tag{7.10}$$

При переході від задачі (7.1) до задачі (7.7)–(7.10) значення двоїстої оцінки (7.2)–(7.3) залишається незмінним.

Запишемо функцію Лагранжа для задачі (7.7)–(7.10):

$$L(x,y,u,v) = x^T A_0 x + b_0{}^T xy + \sum_{i=1}^{m} u_i (x^T A_i x + b_i^T xy + c_i y^2) + v(y^2 - 1),$$

де $u_i \ge 0,\ i \in I^{LQ}$. Для неї умова (7.4) теореми 7.1 про точну двоїсту оцінку матиме такий вигляд: існують u^* і v^*, такі, що

$$\begin{pmatrix} x \\ y \end{pmatrix}^T \begin{pmatrix} A_0 & b_0/2 \\ b_0^T/2 & v^* \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} x \\ y \end{pmatrix}^T \sum_{i=1}^{m} u_i^* \begin{pmatrix} A_i & b_i/2 \\ b_i^T/2 & c_i \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} - v^* - f^* = \sum_{i=1}^{k} l_i^2(x,y).$$

Іншими словами, оцінка є точною тоді і тільки тоді, коли для деякого v^*, по-перше, матриця $A_v(v^*) = \begin{pmatrix} A_0 & b_0/2 \\ b_0^T/2 & v^* \end{pmatrix}$ може бути представлена у вигляді різниці невід'ємно визначеної матриці (відповідає сумі квадратів у правій

частині рівності) і лінійної комбінації $A_u(u^*) = \sum_{i=1}^{m} u_i^* \overline{A}_i$ (за умови $u_i \geq 0$, $i \in I^{LQ}$),

де $\overline{A}_i = \begin{pmatrix} A_i & b_i/2 \\ b_i^T/2 & c_i \end{pmatrix}$, і, по-друге, $v^* + f^* = 0$. Доведення теореми завершено.

Наведемо приклад застосування теореми 7.3.

Приклад 3. Задача про комплементарні власні числа. Задача про комплементарні власні числа (Eigenvalue Complementarity Problem) полягає в такому: задано матриці $A \in R^{n \times n}$ й $B \in R^{n \times n}$; потрібно знайти $\lambda > 0$, $\lambda \in R^1$ і $x \neq 0$, $x \in R^n$, такі, що

$$(\lambda B - A)x \geq 0, \tag{7.11}$$

$$x \geq 0, \tag{7.12}$$

$$x^T(\lambda B - A)x = 0. \tag{7.13}$$

Якщо матриці A і B належать множині симетричних матриць S, задачу про комплементарні власні числа називають симетричною. У роботі [6] розглянутий саме цей випадок при додатковій умові на додатну визначеність матриці B, тобто, коли

$$A, B \in S,\ B \succ 0. \tag{7.14}$$

При виконанні умови (7.14) задача (7.11)–(7.13) зводиться до знаходження стаціонарних точок узагальненого відношення Релея на симплексі [6, с. 1854]:

$$\max_{x \in R^n} \frac{x^T Ax}{x^T Bx}, \tag{7.15}$$

$$x \geq 0, \tag{7.16}$$

$$e^T x = 1, \tag{7.17}$$

де $e := (1,...,1)^T \in R^n$. Стаціонарні точки x^*, що задовольняють нерівності

$$x^{*T} Ax^* > 0, \tag{7.18}$$

є розв'язком задачі (7.11)–(7.14). Це випливає з вимоги додатності параметра λ й того, що матриця B додатно визначена, тобто $x^T Bx > 0$ для усіх x, крім нульового вектора. При цьому $\lambda^* = \frac{x^{*T} Ax^*}{x^{*T} Bx^*}$.

Запропонуємо інше формулювання задачі (7.11)–(7.14). Легко показати, що вона зводиться до знаходження стаціонарних точок квадратичної екстремальної задачі

$$f^* = \min_{x \in R^n}(-x^T Ax), \tag{7.19}$$

$$x \geq 0, \tag{7.20}$$

$$x^T Bx \leq 1. \tag{7.21}$$

Для доведення достатньо зіставити систему Каруша − Куна − Такера для задачі (7.19)–(7.21) та умови початкової задачі (7.11)–(7.14) (відзначимо, що обмеження задачі задовольняють умові регулярності Слейтера).

Розглянемо двоїсту оцінку ψ^* (7.2)–(7.3) для f^* у квадратичній задачі (7.19)–(7.21). Глобальний екстремум задачі (7.19)–(7.21) відповідає розв'язку задачі (7.11)–(7.14) з максимальним комплементарним власним числом $\lambda_{\max}$ (звичайно, якщо розв'язок початкової задачі існує). Якщо $\psi^* = 0$, то задача (7.11)–(7. 14) не має розв'язку; а якщо ні, то можливі різні варіанти.

З теореми 7.3 випливає справедливість наступного твердження.

Твердження 7.1. Для того, щоб двоїста оцінка для задачі (7.19)–(7.21) була точною, необхідно й достатньо, щоб існував вектор $u^* \in \left\{ \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} : u_1 \geq 0, u_1 \in R^n, u_2 \in R^1 \right\}$, для якого виконується

$$\begin{pmatrix} u_2^* B - A & -u_1^*/2 \\ -u_1^{*T}/2 & \lambda_{\max} - u_2^* \end{pmatrix} \succ = 0. \tag{7.22}$$

Зрозуміло, що двоїстий підхід для задачі (7.19)–(7.21) дає точну оцінку тільки при $\lambda_{\max} B - A \succ = 0$. Цю область можна розширити, переформулювавши задачу (7.19)–(7.21) шляхом заміни лінійних обмежень (7.20) квадратичними:

$$f^* = \min_{x \in R^n}(-x^T Ax), \tag{7.23}$$

$$x^T Bx \leq 1, \tag{7.24}$$

$$x_i x_j \geq 0,\ i, j = \overline{1,n}. \tag{7.25}$$

Твердження 7.2. Для того, щоб двоїста оцінка для задачі (7.23)–(7.25) була точною, необхідно й достатньо, щоб виконувалася умова

$$\lambda_{\max} B - A = W + V^+, \tag{7.26}$$

де W — невід'ємно визначена матриця, а V^+ — додатна матриця.

Можна відзначити, що якщо з твердження 7.1 випливає, що при $A \prec= 0$ двоїста оцінка задачі (7.19)–(7.21) дорівнює нулю (так як $\lambda_{\max} = 0$ задовольняє умову (7.22) твердження 7.1 при $u_2^* = 0$), тобто задача (7.11)–(7.14) не має розв'язку, то твердження 7.2 дозволяє підсилити цей результат: якщо матрицю A можна представити у вигляді суми від'ємно визначеної матриці та від'ємної матриці, то задача (7.11)–(7.14) не має розв'язку.

7.4. Достатня умова точної двоїстої оцінки для квадратичної екстремальної задачі

У попередньому розділі сформульована необхідна й достатня умова отримання точної двоїстої оцінки (7.2)–(7.3) для квадратичної екстремальної задачі (7.1) (теорема 7.1). Але поряд із цією умовою, хоча вона й носить загальний характер, хотілося б мати інший критерій, нехай слабший, але який був би зручнішим з погляду застосування на практиці. Нижче сформульована достатня умова, під час виконання якої двоїстий підхід дозволяє знайти значення, а в деяких випадках і точку глобального екстремуму квадратичної екстремальної задачі.

Отже, розглянемо квадратичну екстремальну задачу загального вигляду (7.1) і двоїсту оцінку (7.2)–(7.3) для цієї задачі. Нехай для кожної граничної допустимої точки $u \in (\bar{D} \setminus D) \cap U^+$ задачі (7.2) визначена множина

$$J(u) = \{ j : \lambda_j(u) = 0, j \in \{1,...,n\} \},$$

де $\lambda_j(u), j=\overline{1,n}$, — власні числа матриці $A(u)=A_0+\sum_{i=1}^{m}u_iA_i$ функції Лагранжа задачі (7.1). І нехай $\xi_j(u), j=\overline{1,n}$, — власні вектори, які відповідають цим власним числам $\lambda_j(u)$, $j=\overline{1,n}$. Позначимо x^* і u^* вектори, в яких досягаються значення f^* й ψ^* у задачах (7.1) і (7.2) відповідно.

Теорема 7.4 [7]. Нехай $D\cap U^+\neq\{\varnothing\}$. Якщо існують такий вектор p і таке число $\tilde{\varepsilon}>0$, що для будь-якого $\varepsilon\in(0,\tilde{\varepsilon})$

$$\forall u\in(\overline{D}\setminus D)\cap U^+\ \ \exists j\in J(u)\ \text{ таке, що } \xi_j^T(u)(b_0+\sum_{i=1}^{m}u_ib_i+\varepsilon p)\neq 0, \qquad (7.27)$$

то $\psi^*=f^*$. Причому, якщо умова (7.27) виконується при $p=0$, то вектор $x^*=-A^{-1}(u^*)b(u^*)/2$ розв'язку задачі (7.2)–(7.3) є і розв'язком задачі (7.1).

Доведення. Розглянемо знаходження двоїстої оцінки (7.2)–(7.3) для задачі (7.1). При $u\in D$ внутрішня задача (7.3) строго опукла. Отже, система $L_x'(x,u)=0$ має єдиний розв'язок $x(u)=-A^{-1}(u)b(u)/2$. Це дає можливість записати функцію $\psi(u)$ в явному вигляді

$$\psi(u)=-\frac{1}{4}b^T(u)A^{-1}(u)b(u)+c(u)=$$

$$=-\frac{1}{4}\left(b_0+\sum_{i=1}^{m}u_ib_i\right)^T\left(A_0+\sum_{i=1}^{m}u_iA_i\right)^{-1}\left(b_0+\sum_{i=1}^{m}u_ib_i\right)+\left(c_0+\sum_{i=1}^{m}u_ic_i\right).$$

Функція $\psi(u)$ ввігнута й неперервно диференційовна в області D, а значить і в області визначення $D\bigcap U^+$; її градієнт обчислюється за формулами $\psi_{u_i}'=f_i(x(u)), i=\overline{1,m}$. Дослідимо її, попередньо переписавши таким чином:

$$\psi(u)=-\frac{1}{4}\sum_{j=1}^{n}\left(\xi_j^T(u)(b_0+\sum_{i=1}^{m}u_ib_i)\right)^2/\lambda_j(u)+\left(c_0+\sum_{i=1}^{m}u_ic_i\right) \qquad (7.28)$$

(цей вираз отримано шляхом підстановки замість матриці $A(u)$ функції Лагранжа її представлення за допомогою її власних чисел $\lambda_j(u)$ і власних векторів $\xi_j(u)$, $j=\overline{1,n}$, – $A(u)=A_0+\sum_{i=1}^{m}u_iA_i=\sum_{j=1}^{n}\lambda_j(u)\xi_j(u)\xi_j^T(u)$).

Зрозуміло, що оцінка $\psi^*=\sup\limits_{u\in D\cap U^+}\psi(u)$ (7.28) може досягатися або у внутрішній точці множини D, або коли u прямує до границі додатної визначеності $\bar{D}\cap D$. Вище для кожного $u\in(\bar{D}\setminus D)\cap U^+$ визначена множина $J(u)=\{j:\lambda_j(u)=\min\limits_{l=1,n}\lambda_l(u)=0, j\in\{1,...,n\}\}$, яка відповідає набору індексів дробових членів у виразі (7.28), знаменники яких при даному $u\in(\bar{D}\setminus D)\cap U^+$ стають рівними нулю. Відзначимо, що для внутрішніх точок області визначення (для $u\in D$) ця множина порожня, а на границі невід'ємної визначеності (для $u\in\bar{D}\setminus D$) може складатися з одного або більше елементів, залежно від кратності мінімального власного числа матриці A в точці u ($\lambda_{\min}(u)=\min\limits_{l=1,n}\lambda_l(u)=0$.)

Розглянемо можливі випадки, які можуть виникати залежно від властивостей області визначення двоїстих змінних.

Випадок 1. $\forall u\in(\bar{D}\setminus D)\cap U^+$ $\exists j\in J(u)$ таке, що $\xi_j^T(u)(b_0+\sum_{i=1}^{m}u_ib_i)\neq 0$.

У цьому випадку для усіх значень двоїстих змінних при прямуванні до нуля знаменників дробових членів у виразі (7.28) хоча б один відповідний чисельник не прямує до нуля. Тоді $\psi(u)\to-\infty$ при $u\to\tilde{u}$ для усіх $\tilde{u}\in(\bar{D}\setminus D)\cap U^+$, тобто в усіх граничних точках області додатної визначеності, які належать допустимій області задачі (7.28). У всіх інших точках допустимої області значення функції $\psi(u)$ обмежені. Це означає, що в задачі (7.28) супремум увігнутої функції ($\sup\limits_{u\in D\cap U^+}\psi(u)$) досягається у внутрішній точці області додатної визначеності. Але для цього випадку відомий такий результат.

Лема 7.1 [1, с. 90]. Якщо $\psi^* = \sup\limits_{u \in \bar{D} \cap U^+} \psi(u)$ досягається на множині D, то $\psi^* = f^*$.

Таким чином, у випадку 1, який відповідає умові (7.27) при $p = 0$, оцінка є точною — $\psi^* = f^*$. Крім того, якщо оцінка досягається при деякому $u^* \in D$, то, по-перше, $x(u^*) = -A^{-1}(u^*)b(u^*)/2$ визначається однозначно, і, по-друге, $f_i(x(u^*)) \le 0$, $u_i^* f_i(x(u^*)) = 0$, $i \in I^{LE}$, і $f_i(x(u^*)) = 0$, $i \in I^{EQ}$ (нагадаємо, що градієнт функції $\psi(u)$ при $u \in D$ збігається з вектором нев'язок обмежень задачі (7.1). Отже, $x(u^*) = -A^{-1}(u^*)b(u^*)/2$ — допустима точка задачі (7.1) і є її розв'язком ($f_0(x(u^*)) = \psi^* = f^*$).

Випадок 2. $\exists u \in (\bar{D} \setminus D) \cap U^+$ таке, що $\forall j \in J(u)$ $\xi_j^T(u)(b_0 + \sum\limits_{i=1}^{m} u_i b_i) = 0$.

У цьому випадку існує точка u з області визначення зовнішньої задачі (7.2), в якій якщо знаменник одного з дробових членів у виразі (7.28) дорівнює нулю, то і відповідний чисельник також дорівнює нулю. Тобто на відміну від попереднього випадку 1 існують граничні точки $u \in (\bar{D} \setminus D) \cap U^+$, в яких $\psi(u)$ не прямує до $-\infty$.

Нехай існують такі вектор $p \in R^n$ і додатне число $\tilde{\varepsilon} > 0$, що для будь-якого $\varepsilon \in (0, \tilde{\varepsilon})$

$$\forall u \in (\bar{D} \setminus D) \cap U^+ \ \exists j \in J(u) \text{ таке, що } \xi_j^T(u)(b_0 + \sum_{i=1}^{m} u_i b_i + \varepsilon p) \ne 0.$$

Розглянемо допоміжну задачу, яка відповідає початковій задачі (7.1) зі збуреною цільовою функцією $f_\varepsilon(x) = f_0(x) + \varepsilon(p, x)$:

$$f_\varepsilon^* = f_\varepsilon(x_\varepsilon^*) = \min(f_0(x) + \varepsilon(p, x))$$

при обмеженнях

$$f_i(x) \le 0, i \in I^{LE},$$

$$f_i(x) = 0,\ i \in I^{EQ}.$$

Для цієї задачі маргінальна функція матиме такий вигляд:

$$\psi_\varepsilon(u) = -\frac{1}{4}\sum_{j=1}^{n}\left(\xi_j^T(u)(b_0 + \varepsilon p + \sum_{i=1}^{m} u_i b_i)\right)^2 / \lambda_j(u) + \left(c_0 + \sum_{i=1}^{m} u_i c_i\right).$$

Тоді на підставі умови (7.27) теореми 7.4 для збуреної задачі маємо випадок 1, тобто $\psi_\varepsilon^* = f_\varepsilon^*$. Та оскільки $\left|\psi_\varepsilon^* - \psi^*\right| \to 0$ і $\left|f_\varepsilon^* - f^*\right| \to 0$ при $\varepsilon \to 0$, то $\psi^* = f^*$.

Таки чином доведено, що умова (7.27) є достатньою для того, щоб $\psi^* = f^*$.

Нижче наведемо кілька прикладів практичного застосування достатньої умови отримання точної двоїстої оцінки для квадратичної екстремальної задачі, сформульованої у вигляді теореми 7.4.

Приклад 4. Побудова кулі мінімального радіусу навколо перетину однаково орієнтованих еліпсоїдів. Нехай задано набір однаково орієнтованих еліпсоїдів (тобто з паралельними осями) у просторі R^n:

$$E_i = \{x : (x - d_i)^T A_i (x - d_i) \le 1\},\ i = \overline{1,m},$$

де d_i — центр еліпсоїда, A_i — додатно визначена симетрична матриця. Потрібно знайти кулю мінімального радіусу з фіксованим центром у деякій заданій точці, що містить перетин цього набору еліпсоїдів $\{E_i, i = \overline{1,m}\}$ (припускаємо, що цей перетин не порожній).

Без обмеження загальності далі припускатимемо, що центр шуканої описаної кулі знаходиться у початку координат, а матриці A_i — діагональні ($A_i = diag(a_{ij}, j = 1,\ldots,n)$, де $a_{ij} > 0$, $i = \overline{1,m}$, $j = \overline{1,n}$). Тоді задача, яка тут розглядається може бути представлена у вигляді такої задачі математичного програмування:

$$f^* = \min(-x^T x) \tag{7.29}$$

$$x^T A_i x + b_i^T x + c_i \le 0,\ i = \overline{1,m}, \tag{7.30}$$

де $b_i = -2A_i d_i$, $c_i = d_i^T A_i d_i - 1$, $i = \overline{1,m}$. У цій задачі мінімальне значення цільової функції f^* дорівнює квадрату радіуса шуканої кулі зі знаком мінус.

Для знаходження нижніх оцінок оптимального значення цільової функції неопуклої задачі (7.29)–(7.30) скористаємося двоїстим підходом і, використовуючи теорему 7.4 визначимо випадки, коли для розглянутої задачі цей підхід гарантує знаходження оптимального значення її цільової функції, тобто, коли $\psi^* = f^*$.

Функція Лагранжа задачі (7.29)–(7.30) дорівнює

$$L(x,u) = x^T A(u)x + b^T(u)x + c(u),$$

де $A(u) = diag(-1 + \sum_{i=1}^{m} u_i a_{ij}, j = 1,...,n)$, $b(u) = \sum_{i=1}^{m} u_i b_i$, $c(u) = \sum_{i=1}^{m} u_i c_i$. Усі власні вектори матриці $A(u)$ спрямовані по координатних осях і не залежать від двоїстих змінних, а відповідні їм власні числа дорівнюють $\lambda_j(u) = -1 + \sum_{i=1}^{m} u_i a_{ij}$, $j = \overline{1,n}$. Тоді для задачі (7.29)–(7.30)

$$(\overline{D} \setminus D) \cap U^+ = \left\{ u : \min_{j=1,...,n} (-1 + \sum_{i=1}^{m} u_i a_{ij}) = 0; u \geq 0 \right\}.$$

Шляхом підстановки в нерівність (7.27) при $p = 0$ відповідних значень параметрів задачі (7.29)–(7.30) маємо

$$\xi_j^T(u) b(u) = e_j^T \left(\sum_{i=1}^{m} u_i b_i \right) \neq 0,$$

де $\xi_j(u) = e_j$, $j = \overline{1,n}$, — власні вектори матриці $A(u)$, e_j — n-вимірний вектор, *j-а* компонента якого дорівнює одиниці, а інші дорівнюють нулю.

Таким чином, умова (7.27) при $p = 0$ для задачі (7.29)–(7.30) матиме такий вигляд:

$$\forall u \in \left\{ u : \min_{j=1,...,n} (-1 + \sum_{i=1}^{m} u_i a_{ij}) = 0; u \geq 0 \right\} \ \exists j \in J(u) \text{ таке, що } e_j^T \left(\sum_{i=1}^{m} u_i b_i \right) \neq 0. \quad (7.31)$$

Нехай для деякого $\tilde{u} \in (\overline{D} \setminus D) \cap U^+$ дорівнює нулю $\tilde{j}$-е власне число матриці $A(u)$:

$$\min_{j=1,...,n}(-1+\sum_{i=1}^{m}\tilde{u}_i a_{ij}) = -1+\sum_{i=1}^{m}\tilde{u}_i a_{\tilde{i}\tilde{j}} = 0.$$

Один зі способів задовольнити умову (7.31) — вимагати, щоб нерівність $e_{\tilde{j}}^T\left(\sum_{i=1}^{m}\tilde{u}_i b_i\right) \neq 0$ виконувалася незалежно від значення вектора $\tilde{u}$. Інакше кажучи, щоб $\tilde{j}$-а координата конічної комбінації векторів $\{b_i, i=\overline{1,m}\}$ ніколи не приймала б значення нуль. Це можливо, коли $\tilde{j}$-і координати всіх векторів $b_i, i=\overline{1,m}$, мають один і той самий знак (оскільки за визначенням $\tilde{u} \geq 0$ і нульовий вектор двоїстих змінних не належить області невід'ємної визначеності матриці функції Лагранжа для задачі (7.29)–(7.30), тобто $\tilde{u} \neq 0$). Таким чином, узагальнюючи сказане на всі $\tilde{j} \in \{1,...,n\}$, одержуємо, що умова (7.31) при $p=0$ виконується, якщо всі вектори $b_i, i=\overline{1,m}$, розташовані в одному відкритому ортанті.

Розглянемо випадок, коли серед цих $b_{\tilde{i}\tilde{j}}$, до яких висувається вимога мати однаковий знак, знайдеться $\tilde{i}$, таке, що $b_{\tilde{i}\tilde{j}} = 0$. Тоді достатньо в нульовому векторі p замінити $\tilde{j}$-у компоненту на $p_{\tilde{j}} = sign(\max_{i=1,...,m} b_{\tilde{i}\tilde{j}})$ (або іншими словами, деяке значення зі знаком усіх інших $b_{\tilde{i}\tilde{j}}$, $i \in \{1,...,m\}$, які не дорівнюють нулю), щоб виконалася умова (7.31). У випадку, коли $\forall i \in \{1,...,m\}\ b_{\tilde{i}\tilde{j}} = 0$, координаті $p_{\tilde{j}}$ можна присвоїти будь-яке значення, крім нуля.

Беручи до уваги, що вектор b_i і вектор d_i, що задає центр еліпсоїда $E_i = (A_i, d_i)$, протилежно спрямовані, тому що

$$b_i = -2A_i d_i = -2(a_{i1}d_{i1}\ a_{i2}d_{i2}...a_{in}d_{in})^T,$$

одержуємо справедливість такого твердження.

Твердження 7.3 [8]. Якщо центри всіх еліпсоїдів розташовані в одному закритому ортанті, то двоїста оцінка (7.2)–(7.3) задачі (7.29)–(7.30) точна. Якщо ж вони розташовані в одному відкритому ортанті, то в результаті знаходження двоїстої оцінки одержуємо також і точку розв'язку початкової задачі $x^* = -A^{-1}(u^*)b(u^*)/2$.

Окремий випадок задачі (7.29)–(7.30) для гомотетичних еліпсоїдів. Розглянемо задачу побудови кулі мінімального об'єму із заданим центром, описаної навколо перетину гомотетичних еліпсоїдів (тобто з однаковими з точністю до додатного множника матрицями у відповідних їм квадратичних формах). Для цього часткового випадку задачі (7.29)–(7.30) постановка задачі математичного програмування спроститься:

$$f^* = \min(-x^T x) \tag{7.32}$$

при обмеженнях

$$x^T A_1 x + b_i^T x + c_i \le 0,\ i = \overline{1,m}. \tag{7.33}$$

У задачі (7.32)–(7.33), як і в задачі (7.29)–(7.30), без обмеження загальності вважатимемо, що центр описаної кулі знаходить у центрі координат, а $A_1 = diag(a_j, j = 1,\ldots,n)$ є діагональною матрицею.

Функція Лагранжа для квадратичної екстремальної задачі (7.32)–(7.33) дорівнює $L(x,u) = x^T A(u)x + b^T(u)x + c(u)$, де

$$A(u) = diag(-1 + a_j \sum_{i=1}^{m} u_i, j = 1,...,n),\ b(u) = \sum_{i=1}^{m} u_i b_i,\ c(u) = \sum_{i=1}^{m} u_i c_i.$$

Власні вектори матриці $A(u)$ спрямовані по координатних осях і не залежать від двоїстих змінних, а власні числа визначаються як

$$\lambda_j(u) = -1 + a_j \sum_{i=1}^{m} u_i,\ j = \overline{1,n}.$$

Для задачі (7.32)–(7.33)

$$(\bar{D} \setminus D) \cap U^+ = \left\{ u : \min_{j=1,\ldots,n} (-1 + a_j \sum_{i=1}^{m} u_i) = 0; u \ge 0 \right\} =$$

$$= \left\{ u : (-1 + \left(\min_{j=1,\ldots,n} a_j \right) \sum_{i=1}^{m} u_i = 0; u \geq 0 \right\} = \left\{ u : (-1 + a_{j\min} \sum_{i=1}^{m} u_i = 0; u \geq 0 \right\},$$

а множина індексів $J(u)$ для всіх $u \in (\bar{D} \setminus D) \cap U^+$ однакова й складається з єдиного елемента $j\min$, що відповідає мінімальному власному числу матриці A_1. Тоді умова (7.27) при $p = 0$ для задачі (7.32)–(7.33) має вигляд:

$$\forall u \in \left\{ u : (-1 + a_{j\min} \sum_{i=1}^{m} u_i = 0; u \geq 0 \right\} \quad \xi_{j\min}^T(u)(b_0 + \sum_{i=1}^{m} u_i b_i) = (\sum_{i=1}^{m} u_i b_i)_{j\min} \neq 0,$$

яка гарантовано виконується, якщо $j\min$-і компоненти всіх векторів $b_i, i = \overline{1,m}$, одного знаку.

У випадку, коли серед $b_{ij\min}$ знайдеться $\tilde{i}$, таке, що $b_{\tilde{i}j\min} = 0$, достатньо в нульовому векторі p замінити $j\min$-ю компоненту на $p_{j\min} = sign(\max_{i=1,\ldots,m} b_{ij\min})$, щоб виконалася умова (7.27) теореми 7.4.

Твердження 7.4 [8]. Якщо координати центрів усіх еліпсоїдів, які відповідають мінімальному власному числу матриці A_1, мають один знак або дорівнюють нулю, то двоїста оцінка (7.2)–(7.3) задачі (7.32)–(7.33) точна. Якщо ж вони строго одного знаку, то в результаті знаходження двоїстої оцінки одержуємо також і точку розв'язку початкової задачі $x^* = -A^{-1}(u^*)b(u^*)/2$.

Окремий випадок задачі (7.29)–(7.30) для куль. Розглянемо відому задачу побудови кулі мінімального об'єму із заданим центром у центрі координат, описаної навколо перетину заданих куль:

$$f^* = \min(-x^T x) \tag{7.34}$$

при обмеженнях

$$x^T x + b_i^T x + c_i \leq 0, \; i = \overline{1,m}. \tag{7.35}$$

У цій задачі матриця $A(u) = diag(-1 + \sum_{i=1}^{m} u_i, j = 1,\ldots,n)$ функції Лагранжа має власні вектори, спрямовані по координатних осях, і всі її власні числа однакові:

$$\lambda_j(u) = -1 + \sum_{i=1}^{m} u_i\,,\ j = \overline{1,n}\,.$$

Умова (7.27) при $p = 0$ для задачі (7.34)–(7.35) має такий вигляд:

$$\forall u \in \left\{ u : (-1 + \sum_{i=1}^{m} u_i = 0; u \geq 0 \right\} \quad \sum_{i=1}^{m} u_i b_i \neq 0\,,$$

що означає $0 \notin \text{int}(conv\{b_i, i = 1,...,m\})$. У випадку, коли центр координат лежить на границі $conv\{b_i, i = 1,...,m\}$, достатньо прийняти $p = \sum_{i=1}^{m} u_i b_i$ при $u > 0$, щоб виконалася умова (7.27) теореми 7.4.

Твердження 7.5 [8]. Якщо $0 \notin \text{int}(conv\{b_i, i = 1,...,m\})$, то двоїста оцінка задачі (7.34)–(7.35) точна. Якщо ж $0 \notin conv\{b_i, i = 1,...,m\}$, то в результаті знаходження двоїстої оцінки одержуємо також і точку розв'язку початкової задачі $x^* = -A^{-1}(u^*)b(u^*)/2$.

Відзначимо, що для випадку $0 \in \text{int}(co\{b_i, i = 1,...,m\})$, можна навести простий приклад, коли двоїста оцінка не є точною — у квадратичній задачі від однієї змінної

$$f^* = \min(-x^2)$$

при обмеженнях

$$(x+2)^2 \leq 25\,,$$

$$(x-5)^2 \leq 9\,,$$

для якої $f^* = f(3) = -9$, двоїста оцінка $\psi^* = -10.42$ (досягається при $u^* = (0.7143, 0.2857)$, $x^* = 2.8289$). Тобто $\psi^* \neq f^*$.

Приклад 5. Спеціальна задача, пов'язана з визначенням інваріантних множин динамічних систем. При синтезі керування, що мінімізує область локалізації інваріантної множини сімейства нелінійних систем, які піддаються впливу обмежених збурювань, може виникнути необхідність у розв'язуванні такої задачі [9]:

$$\min_{x\in R^n}(d^T x + k\|x\|) \tag{7.36}$$

при обмеженні

$$(x - x_0)^T Q(x - x_0) \le 1, \tag{7.37}$$

де задані симетрична додатно визначена матриця Q розмірності $n \times n$, n-вимірні вектори d і x_0, скаляр k; $\|x\| = \sqrt{\sum_{i=1}^{n} x_i^2}$ — довжина вектора $x = (x_1, x_2, \ldots, x_n)^T \in R^n$. У випадку $k \ge 0$ задача (7.36)–(7.37) є задачею опуклого програмування й відноситься до спеціального класу «second-order cone programming problems» [10, 11], який є достатньо дослідженим (наприклад у [10] зазначені деякі відомі «вирішувачі», що реалізують, як правило, модифікації методу внутрішніх точок: AMPL, CPLEX, ECOS, Joptimizer, OpenOpt, SDPT3 та інші). Більший інтерес викликає випадок $k < 0$, коли задача неопукла. Нижче пропонується шлях розв'язування цієї задачі на основі її формулювання у вигляді квадратичної екстремальної задачі й застосуванні двоїстого підходу [1, 2] для знаходження нижньої оцінки оптимального значення цільової функції отриманої задачі.

Задача (7.36)–(7.37) для випадку $k < 0$ може бути записана у вигляді квадратичної оптимізаційної задачі

$$f^* = f_0(x^*) = \min_{x\in R^n, y\in R^1}(d^T x + ky) \tag{7.38}$$

при обмеженнях

$$(x - x_0)^T Q(x - x_0) = 1, \tag{7.39}$$

$$y^2 = x^T x. \tag{7.40}$$

Відзначимо, що змінна y, яка відповідає нормі вектора x, є величиною додатною. Однак включати відповідне обмеження у постановку задачі (7.38)–(7.40) немає необхідності, тому що воно виконується автоматично. Це легко випливає з очевидного співвідношення

$$f_0(x, \overline{y}) = d^T x + k\overline{y} \le f_0(x, -\overline{y}) = d^T x + k(-\overline{y}),$$

де $k<0$, а $\overline{y}\geq 0$ — довільне значення змінної y з області визначення задачі (7.38)–(7.40).

Розглянемо двоїсту оцінку (7.2)–(7.3) для квадратичних екстремальних задач для оцінки знизу оптимального значення f^* цільової функції в задачі (7.38)–(7.40).

Позначимо вектор змінних задачі (7.38)–(7.40) як $z=\begin{pmatrix} x \\ y \end{pmatrix}\in R^{n+1}$. Для цієї квадратичної задачі двоїста оцінка (7.2)–(7.3) має такий вигляд:

$$\psi_1^*=\sup_{u\in D}\left(\inf_{z\in R^n}\left(z^T A_1(u)z+b_1^T(u)z+c_1(u)\right)\right)\leq f^*, \tag{7.41}$$

де

$$A_1(u)=\begin{pmatrix} u_1Q-u_2I & 0 \\ 0 & u_2 \end{pmatrix},\ b_1(u)=\begin{pmatrix} d-2u_1Qx_0 \\ k \end{pmatrix},\ \mathrm{c}_1(u)=u_1(x_0^TQx_0-1).$$

Скористаємося теоремою 7.4 для того, щоб сформулювати достатню умову отримання точної нижньої оцінки ψ_1^* (7.41) для задачі (7.38)–(7.40) (квадратичний аналог досліджуваної задачі (7.36)–(7.37)), попередньо представивши її у зручнішій для цієї мети формі.

Для симетричної дійсної матриці Q має місце розкладання $Q=Udiag(\beta)U^T=\sum_{j=1}^{n}\beta_j\eta_j\eta_j^T$, де $\beta=(\beta_1\beta_2\ldots\beta_n)^T$ — вектор її власних чисел, і $U=(\eta_1\eta_2\ldots\eta_n)$ — матриця, стовпцями якої є її власні вектори η_j, $j=\overline{1,n}$. Позначимо $J_{\min}$ — множину індексів власних чисел, які дорівнюють мінімальному власному числу $\beta_{\min}=\min_{j=1,n}(\beta_j)$ матриці Q; відповідні цим однаковим власним числам власні вектори утворюють множину $\{\eta_j\}_{j\in J_{\min}}$.

Шляхом заміни змінних $x=U\tilde{x}$ у задачі (7.38)–(7.40) отримуємо задачу

$$f^*=\min_{\tilde{x}\in R^n,\,y\in R^1}\left(\left(U^Td\right)^T\tilde{x}+ky\right) \tag{7.42}$$

при обмеженнях

$$(\tilde{x} - U^T x_0)^T diag(\beta)(\tilde{x} - U^T x_0) = 1, \tag{7.43}$$

$$y^2 - \tilde{x}^T \tilde{x} = 0. \tag{7.44}$$

Позначимо вектор змінних цієї задачі $\tilde{z} = \begin{pmatrix} \tilde{x} \\ y \end{pmatrix} \in R^{n+1}$. Двоїста оцінка (7.2)–(7.3) для квадратичної задачі (7.42)–(7.44) дорівнює

$$\psi_2^* = \sup_{u \in D} \left(\inf_{\tilde{z} \in R^n} \left(\tilde{z}^T A_2(u) \tilde{z} + b_2^T(u) \tilde{z} + c_2(u) \right) \right) \le f^*, \tag{7.45}$$

де

$$A_2(u) = \begin{pmatrix} u_1 diag(\beta) - u_2 I & 0 \\ 0 & u_2 \end{pmatrix},\ b_2(u) = \begin{pmatrix} U^T d - 2u_1 diag(\beta) U^T x_0 \\ k \end{pmatrix},$$

$$c_2(u) = u_1(x_0^T Q x_0 - 1),\ I \text{ — одинична матриця.}$$

Оскільки невироджене лінійне перетворення простору (у цьому випадку $\begin{pmatrix} \tilde{x} \\ y \end{pmatrix} = \begin{pmatrix} U^T & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$) не впливає на значення двоїстої оцінки, двоїсті оцінки ψ_1^* (7.41) для задачі (7.38)–(7.40) і ψ_2^* (7.45) для задачі (7.42)–(7.44) співпадають — $\psi_1^* = \psi_2^*$. Також не змінюється (з урахуванням перетворення простору) і множина точок, на яких це значення досягається.

Твердження 7.6 [7]. Якщо система

$$\begin{cases} \eta_j^T (d - 2u_1 \beta_{\min} x_0) = 0,\ j \in J_{\min} \\ u_1 > 0 \end{cases} \tag{7.46}$$

несумісна, то розв'язок задачі (7.45) дає точку глобального екстремуму задачі (7.42)–(7.44) і $\psi_2^* = f^*$.

Доведення. Розглянемо умову (7.27) теореми 7.4 при $p = 0$. Оскільки в задачі (7.45)

$$D = \left\{ u : \min_{j=1,n} (u_1 \beta_j - u_2) > 0, u_2 > 0 \right\} = \left\{ u : (u_1 \beta_{\min} - u_2) > 0, u_2 > 0 \right\},$$

множина $(\bar{D} \setminus D) = \{u : (u_1\beta_{\min} - u_2) = 0\}$ (змінна u_2 завжди додатна, інакше $\psi_2^* = -\infty$) і $\forall u \in (\bar{D} \setminus D)\ \ J(u) = J_{\min}$. Більш наочно це видно з виразу двоїстої функції для задачі (7.42)–(7.44):

$$\psi_2(u) = -\frac{1}{4}\sum_{j=1}^{n}\left(U^T d - 2u_1\beta_j U^T x_0\right)_j^2 / \left(u_1\beta_j - u_2\right) - k^2 / u_2 + u_1\left(x_0^T Q x_0 - 1\right) =$$

$$= -\frac{1}{4}\sum_{j=1}^{n}\left(\eta_j^T (d - 2u_1\beta_j x_0)\right)^2 / \left(u_1\beta_j - u_2\right) - k^2 / u_2 + u_1\left(x_0^T Q x_0 - 1\right),$$

звідки очевидно, що нульові знаменники на границі області $\bar{D}$ можуть бути лише в дробових доданках функції $\psi_2(u)$, які відповідають мінімальному числу $\beta_{\min}$.

Шляхом підстановки в нерівність умови (7.27) теореми 7.4 при $p = 0$ відповідних значень параметрів задачі (7.45):

$$\xi_j^T(u) b_2(u) = e_j^T \begin{pmatrix} U^T d - 2u_1 diag(\beta) U^T x_0 \\ k \end{pmatrix} = \begin{cases} \eta_j^T (d - 2u_1\beta_{\min} x_0), & if\ \ j = 1,...,n \\ k, & if\ \ j = n+1 \end{cases},$$

де $\xi_j(u) = e_j$ — власні вектори матриці $A_2(u)$ (e_j — вектор, *j-a* компонента якого дорівнює одиниці, а інші дорівнюють нулю), отримуємо

$$\xi_j^T(u)(b_0 + \sum_{i=1}^{m} u_i b_i) = \xi_j^T(u) b_2(u) = \eta_j^T (d - 2u_1\beta_{\min} x_0) \neq 0.$$

Таким чином, умова (7.27) теореми 7.4 при $p = 0$ для задачі (7.42)–(7.44) матиме такий вигляд:

$$\forall u \in (\bar{D} \setminus D)\ \ \exists j \in J_{\min} \text{ таке, що } \eta_j^T (d - 2u_1\beta_{\min} x_0) \neq 0,$$

або, враховуючи, що $(\bar{D} \setminus D) = \{u : (u_1\beta_{\min} - u_2) = 0\}$,

$$\forall u_1 > 0\ \ \exists j \in J_{\min} \text{ таке, що } \eta_j^T (d - 2u_1\beta_{\min} x_0) \neq 0. \qquad (7.47)$$

Беручи до уваги, що отримана умова (7.47) еквівалентна несумісності системи від однієї змінної

$$\begin{cases} \eta_j^T (d - 2u_1\beta_{\min} x_0) = 0,\ j \in J_{\min} \\ u_1 > 0 \end{cases},$$

доведення твердження 7.6 завершено.

З доведеного твердження випливає досить проста для практичного використання процедура перевірки на точність двоїстої оцінки для конкретних задач виду (7.42)–(7.44): з першої рівності системи (7.46) знаходимо $\bar{u}_1$ і, якщо воно додатне, підставляємо в інші рівності. Перше ж порушення рівності означає, що розв'язування задачі (7.45) дає оптимальне значення і точку глобального мінімуму задачі (7.42)–(7.44); якщо ж система виявиться сумісною, то оцінка, одержана в результаті розв'язання задачі (7.45), може бути як рівною, так і менше глобального мінімуму задачі (7.42)–(7.44).

У частковому випадку, коли кратність мінімального власного числа матриці Q дорівнює одиниці ($|J_{\min}| = 1$), твердження 7.6 може бути переформульовано у вигляді твердження 7.7.

Твердження 7.7 [7]. Якщо множина $J_{\min}$ складається з єдиного індексу s ($s \in \{1,2,\ldots,n\}$) і числа $\eta_s^T d$ й $\eta_s^T x_0$ мають протилежні знаки, то розв'язок задачі (7.45) дає точку глобального максимуму задачі (7.42)–(7.44) і $\psi_2^* = f^*$.

Цьому твердженню може бути дана така геометрична інтерпретація: двоїста оцінка буде точною, якщо вектори d й x_0 лежать у різних півпросторах, утворених гіперплощиною, нормаллю якої є власний вектор η_s матриці Q, що відповідає її єдиному мінімальному власному числу, і яка проходить через початок координат. Або, інакше кажучи, $\cos(\eta_s, d)\cos(\eta_s, x_0) < 0$.

На закінчення акцентуємо увагу на тому, що оскільки, як відзначалося вище, задачі (7.36)–(7.37), (7.38)–(7.40) і (7.42)–(7.44) еквівалентні, а двоїсті оцінки (7.41) і (7.45) для відповідних квадратичних задач співпадають, то результати, сформульовані у вигляді тверджень 7.4 і 7.6, слушні також для задач (7.36)–(7.37) і (7.38)–(7.40).

7.5. Висновки

Застосування лагранжевої релаксації для розв'язування квадратичних екстремальних задач приводить до необхідності дослідження якості отриманих при цьому оцінок. Особливий інтерес викликають випадки, коли цей підхід дозволяє знайти глобальний екстремум початкової задачі. У цій роботі зроблені певні кроки в цьому напрямку, а саме наведено необхідну та достатню умову (теореми 7.1, 7.3) та достатню умову (теорема 7.4), при виконанні яких значення глобального екстремуму квадратичної екстремальної задачі та значення її двоїстої оцінки співпадають. Використання цих результатів нерідко може надати суттєву допомогу при дослідженні конкретних задач на можливість отримання їх розв'язків шляхом знаходження двоїстої оцінки. Для прикладу, у роботі розглянуто декілька задач, для яких побудовано еквівалентні квадратичні екстремальні задачі та вказано випадки, коли знаходження двоїстих оцінок для цих задач гарантує розв'язання початкової задачі (приклади 1–5).

Список літератури

1. Шор Н. З., Стеценко С. И. Квадратичные экстремальные задачи и недифференцируемая оптимизация. Киев: Наукова думка, 1989. 208 с.
2. Shor N. Z. Nondifferentiable Optimization and Polynomial Problems. Dordrecht, Kluwer, 1998. 394 p.
3. Anstreicher K., Wolkowicz H. On Lagrangian relaxation of quadratic matrix constraints. SIAM Journal on Matrix Analysis and Applications. 2000. 22 (1). С. 41–55.
4. Березовский О. А. О точности двойственных оценок для квадратичных экстремальных задач. Кибернетика и системный анализ. 2012. № 1. С. 33–39.

5. Березовский О. А. Условие точности двойственных квадратичных оценок в матричном виде. Теорія оптимальних рішень. Київ: Інститут кібернетики ім. В. М. Глушкова НАН України, 2015. С. 41–45.
6. Queiroz M., Júdice J.J., Humes Jr.C. The symmetric eigenvalue complementarity problem. Mathematics of Computation. 2004. V. 73. № 248. С. 1849–1863.
7. Березовский О. А. О решении одной специальной оптимизационной задачи, связанной с определением инвариантных множеств динамических систем. Проблемы управления и информатики. 2015. № 3. С. 33–40.
8. Березовский О. А., Шулинок И. Э. Использование двойственного подхода для решения одной геометрической задачи. Компьютерная математика. Киев: Институт кибернетики им. В. М. Глушкова НАН Украины, 2016. № 2. С. 94–99.
9. Кунцевич А. В., Кунцевич В. М. Синтез управления инвариантными множествами семейств линейных и нелинейных дискретных систем с ограниченными возмущениями. Кибернетика и системный анализ. 2011. № 4. С. 65–78.
10. Second-order cone programming. Wikipedia. http://en.wikipedia.org/wiki/Second-order_cone_programming.
11. Alizadeh F., Goldfarb D. Second-order cone programming. Mathematical programming. 2003. 95 (1). P. 3–51.

8. ПРИНЦИП РОЗТЯГУВАННЯ ЧАСУ ПРИ ПРИЙНЯТТІ РІШЕНЬ В УМОВАХ КОНФЛІКТУ ТА НЕВИЗНАЧЕНОСТІ

Г. Ц. Чикрій

Анотація. Розвинутий підхід до вирішення ігрових задач зближення, для яких не виконана класична умова Л. С. Понтрягіна переваги переслідувача над втікачем у ресурсах керування. Він полягає в переході від вихідної гри з повною інформацією до гри зі змінним запізненням інформації, яка виражена через функцію розтягування часу і зникає в кінці гри, та подальшому аналізі останньої на основі її еквівалентності певній грі з повною інформацією. Результати проілюстровані на модельних прикладах м'якої зустрічі керованих систем другого порядку.

Annotation. The paper concerns game problems of pursuit for which classic Pontryagin's condition of the pursuer advantage over the evader in control resources does not hold. An approach for solving such problems is developed that consists in transition from original game with complete information to the auxiliary game with variable information delay and subsequent analysis of the latter on the basis of its equivalence to certain game with complete information. In so doing, the function of information delay is expressed through the function of time stretching and turns into zero at the instant of game termination. The results are exemplified with the model examples of soft meeting of controlled second-order systems.

8.1. Вступ

Загальна теорія диференціальних ігор, як правило, передбачає повну інформацію про стан конфліктуючих сторін. Так завжди було в класичних роботах перших номерів світового рейтингу в галузі конфліктно-керованих процесів Л. С. Понтрягіна [1] та М. М. Красовського. Проте на практиці важко розраховувати на подібні ідеальні умови. Завжди присутні або неточне вимірювання стану, або запізнення інформації у часі. Остання ситуація видається найприроднішою. При аналізі ситуації із запізненням в основі лежить ідея, що конфліктно-керований процес із запізненням зводиться до еквівалент-

ного процесу з повною інформацією та іншою термінальною множиною [2]. До аналізу останнього можуть бути застосовані класичні методи теорії диференціальних ігор. Зауважимо, що окремо вивчені випадки постійного запізнення інформації та запізнення, що залежить від часу або фазового стану.

При застосуванні першого прямого методу Понтрягіна [1], одного з найефективніших та найпростіших методів за своєю структурою, ключовою умовою — умовою переваги переслідувача над втікачем, є умова Понтрягіна. Для широкого кола задач, зокрема для протидіючих сторін, динаміка яких описується рівняннями з різною інерційністю або коливними системами, у задачах про м'яку зустріч ця умова не виконується на певних інтервалах часу [3].

Результатом глибокого вивчення цієї умови [4] була її модифікація [5], що передбачає можливість побудови керування переслідувача з керування втікача в минулому. Встановлення зв'язку цієї модифікації з переходом до гри зі змінним запізненням інформації [6] стимулювало розвиток нового підходу, пов'язаного з розтягуванням часу, який дозволяє забезпечити виконання певного аналога умови Понтрягіна та привести траєкторію процесу на задану термінальну множину дещо пізніше. Проте мета — закінчення гри за скінчений час все ж буде досягнута.

Зміст цього підходу полягає в штучному погіршенні інформаційних можливостей переслідувача, а саме в переході від вихідної гри з повною інформацією до гри з тією самою динамікою і термінальною множиною, але із запізненням інформації, що зменшується у процесі наближення траєкторії гри до термінальної множини та перетворюється у нуль при попаданні на неї.

Далі одержана гра із запізненням інформації аналізується на основі її еквівалентності грі з повною інформацією, для якої умова Л. С. Понтрягіна вже містить функцію розтягування часу. Введена числова функцію $I(t)$ за змістом є функцією розтягування часу.

Схему ігрового підходу, в якому модифікована умова Понтрягіна використовує функцію $I(t)$, називатимемо принципом розтягування часу. Його

використання дозволило провести повний аналіз задачі про м'яку зустріч диференційних систем другого порядку, що описують динаміку відповідно до 2-го закону Ньютона за наявності тертя [7]. Отримані достатні умови на параметри гри, які забезпечують переслідувачу можливість здійснення за скінчений час м'якої зустрічі об'єктів. Окремо встановлено умови на початкові стани об'єктів, при яких переслідування здійснюється строго вздовж сліду втікача.

Аналогічні результати отримані для задачі про м'яку зустріч керованих коливних систем другого порядку, що описують динаміку математичного маятника, затухаючих коливань та різнотипних об'єктів [8].

Зауважимо, що ефект запізнення (еквівалентність гри з запізненням грі з повною інформацією) та принцип розтягування часу працюють і у випадку динамічної гри загального вигляду [9, 10]. Представлення траєкторії такої гри охоплює широке коло конфліктно-керованих процесів, а саме процесів, динаміка яких задається системами звичайних диференціальних, диференціально-різницевих, інтегральних, інтегродиференціальних рівнянь та системами з дробовими похідними [11, 12].

8.2. Ефект запізнення інформації в лінійних диференціальних іграх

Нехай динаміка конфліктно-керованого процесу описується системою лінійних диференціальних рівнянь

$$\dot{z} = Az + u + v, \quad z(0) = z_0. \tag{8.1}$$

Тут $z \in R^n$, $A-$ квадратна матриця порядку n, u і v — керування, які обидва гравці в кожний момент часу обирають з областей U та V таким чином, щоб їх реалізації у часі були вимірними за Лебегом функціями, причому $U \in K\left(R^n\right)$, $V \in K\left(R^n\right)$. Позначимо Ω_U, Ω_V як відповідні множини допустимих керувань.

Задана замкнена термінальна множина M_*, $M_* \subset R^n$.

Гра розглядається з погляду переслідувача, мета якого привести траєкторію системи (8.1) не пізніше ніж/або точно в скінчений момент часу T на термінальну множину M_* за будь-якої протидії втікача, а саме: $z(T) \in M_*$.

Задача, що стоїть перед переслідувачем, ускладнена тим, що інформація про поточний стан процесу стає відомою переслідувачу із запізненням у часі. Це запізнення є функцією часу $\tau(t)$, що визначена на інтервалі $[\tau_0, +\infty)$, де $\tau_0 > 0$. $\tau(\tau_0) = \tau_0$. Оскільки гра (8.1) проходить при $t \geq 0$ і в момент t стає відомою інформація про стан системи в момент $t - \tau(t)$, то має виконуватись нерівність $t - \tau(t) \geq 0$ або $\tau(t) \leq t$, $t \geq \tau_0$. Вважатимемо, що $\tau(t)$ — додатна кусково-неперервна функція, яка може мати лише зліченну кількість розривів першого роду та є абсолютно неперервною в області своєї неперервності, причому в точках, де існує похідна $\tau(t)$, вона задовольняє наступному обмеженню на швидкість росту: $\dot{\tau}(t) \leq 1$. Остання умова забезпечує те, що змінна $t - \tau(t)$ не спадає, тобто доступна інформація про траєкторію конфліктно-керованого процесу з часом поповнюється.

Гра починається у момент $t = 0$, але перша інформація про стан гри доходить до першого гравця (переслідувача) в момент часу τ_0. Отже, на початковому відрізку часу $[0, \tau_0)$ переслідувач використовує довільне допустиме керування, позначимо його $u^{\tau_0}(\cdot)$, $u^{\tau_0}(\cdot) = \{u(s) : s \in [0, \tau_0)\}$. У момент τ_0 τ_0 переслідувач дізнається про початковий стан об'єкта z_0, також він пам'ятає власне керування на відрізку $[0, \tau_0)$. Пару $(z_0, u^{\tau_0}(\cdot))$ називатимемо початковою позицією гри. У момент часу $t, t \geq 0,$ переслідувачу стає відомим стан процесу в момент часу $t - \tau(t)$, а саме $z(t - \tau(t))$. Пару $(z(t - \tau(t)), u^t(\cdot))$, де $u^t(\cdot) = \{u(s) : s \in [t - \tau(t), t)\}$ — реалізація керування переслідувача на інтервалі $[t - \tau(t), t)$, називатимемо позицією гри в момент часу t.

З огляду на інформованість переслідувача сукупність усіх можливих станів процесу в момент t за формулою Коші описується множиною

$$Z(t)=\left\{z: z=e^{tA}z_0+\int_0^t e^{(t-\theta)A}u(\theta)d\theta+\right.$$

$$\left.+\int_0^{t-\tau(t)} e^{(t-\theta)A}v(\theta)d\theta+\int_{t-\tau(t)}^{t} e^{(t-\theta)A}v_1(\theta)d\theta, v_1(\cdot)\in\Omega_V\right\}.$$

Будемо казати, що, починаючи з початкової позиції $\left(z_0, u^{\tau_0}(\cdot)\right)$, гра може бути закінчена в момент часу t_1, якщо переслідувач на основі доступної йому інформації шляхом вибору свого керування може досягти включення $Z(t_1)\subset M_*$ за будь-якої поведінки втікача.

Введемо багатозначне відображення

$$V(\tau(t))=\left\{x: x=\int_{t-\tau(t)}^{t} e^{(t-\theta)A}v(\theta)d\theta, v(\cdot)\in\Omega_V\right\}.$$

Бачимо, що $V(\tau(t))$ це сукупність значень інтеграла при всіх можливих реалізаціях керування втікача $v(\theta)$ на відрізку $\left[t-\tau(t), t\right)$ із значеннями в області керування V. Зробивши заміну $t-\theta=\theta_1$ під інтегралом у виразі для $V(\tau(t))$, приведемо його до вигляду:

$$V(\tau(t))=\left\{x: x=\int_0^{\tau(t)} e^{\theta A}v(\theta)d\theta, v(\cdot)\in\Omega_V\right\}. \tag{8.2}$$

Позначимо

$$\tilde{z}(t)=e^{tA}z_0+\int_0^t e^{(t-\theta)A}u(\theta)d\theta+\int_0^{t-\tau(t)} e^{(t-\theta)A}v(\theta)d\theta. \tag{8.3}$$

Тоді множина $Z(t)$ може бути представлена у вигляді

$$Z(t)=\tilde{z}(t)+V(\tau(t)).$$

Нова змінна $\tilde{z}(t)$ майже всюди на півосі $[\tau_0,+\infty)$ задовольняє таке диференціальне рівняння:

$$\dot{\tilde{z}}(t) = A\,\tilde{z}(t) + u(t) + \left(1-\dot{\tau}(t)\right)e^{\tau(t)A}\,v\left(t-\tau(t)\right) \tag{8.4}$$

з початковою умовою

$$\tilde{z}_0 = \tilde{z}(\tau_0) = e^{\tau_0 A}z_0 + \int_0^{\tau_0} e^{(\tau_0-\theta)A}u(\theta)\,d\theta. \tag{8.5}$$

У подальшому в нагоді стане операція геометричної різниці Мінковського [1]:

$$X \underline{*} Y = \{z : z + Y \subset X\} = \bigcap_{y\in Y}(X-y), \quad X\in R^n,\ Y\in R^n.$$

Позначимо

$$M_*(t) = M_* \underline{*} V(\tau(t)) \tag{8.6}$$

і розглянемо диференціальну гру зближення з повною інформацією, в якій рух об'єкта описується системою диференціальних рівнянь (8.4) з початковою умовою (8.5), а термінальною множиною є багатозначне відображення $M_*(t)$ (8.6).

Неважко помітити, що якщо в деякий скінчений момент часу t_1 виповнилося включення $Z(t_1)\subset M_*$, то з цього автоматично випливає, що $\tilde{z}(t_1)\in M_*(t_1)$. І, навпаки, якщо в деякий момент часу $t_1, t_1<\infty$, вектор $\tilde{z}(t_1)$ опинився на множині $M_*(t_1)$, то має місце включення $Z(t_1)\subset M_*$, що, за означенням, свідчить про завершення гри із змінним запізненням інформації. Звідси випливає така теорема.

Теорема 8.1. Нехай багатозначне відображення $M_*(t)$ має непорожні образи для усіх $t\in[\tau_0,+\infty)$. Якщо диференційна гра зближення (8.1) з термінальною множиною M_* і запізненням інформації $\tau(t)$ може бути завершена в момент часу t_1, починаючи із позиції $\left(z_0,u^{\tau_0}(\cdot)\right)$, то відповідна гра зближення з повною інформацією (8.4), (8.5) і термінальною множиною (8.6)

може бути завершена в той самий момент t_1, починаючи з положення $\tilde{z}_0$, і навпаки.

Зауваження 1. У випадку заздалегідь відомого моменту t_1 завершення гри t_1 у теоремі 8.1 достатньо припущення про непорожність множини $M_*(t)$ лише в точці $t = t_1$, а саме $M_*(t_1) \neq \varnothing$.

Теорема 8.1 відкриває можливість застосовувати класичні методи, розроблені для лінійних ігор з повною інформацією, до розв'язання лінійних диференційних ігор зі змінним у часі запізненням інформації про фазовий стан об'єкта. З іншого боку, факт еквівалентності обґрунтовує підхід, пов'язаний з розтягуванням часу, до розв'язку ігор з повною інформацією, для яких не виконана умова переваги Л. С. Понтрягіна.

8.3. Використання ефекту запізнення інформації в лінійних диференціальних іграх зближення з повною інформацією

Нехай динаміка конфліктно-керованого процесу описується системою лінійних диференціальних рівнянь

$$\dot{z} = Az + u - v. \tag{8.7}$$

Гра розглядається з погляду переслідувача, метою якого є приведення траєкторії системи (8.1) на термінальну множину M_0, $M_0 \subset R^n$. Припустимо, що множина M_0 є лінійним підпростором у R^n. Позначимо π оператор ортогонального проектування з R^n на L, де L — ортогональне доповнення до M_0 в R^n. Тоді вихід траєкторії гри на множину M_0 у момент T ($z(T) \in M_0$) еквівалентний співвідношенню $\pi z(T) = 0$.

Для досягнення мети переслідувач використовує контрстратегії, тобто в кожний момент часу будує своє керування на основі знання миттєвого керування супротивника. При цьому гравці вибирають свої керування таким чином, щоб їх реалізації у часі були вимірними функціями.

Перший прямий метод Л. С. Понтрягіна, як і інші прямі методи переслідування з повною інформацією, передбачає, що в процесі гри переслідувач вибирає своє керування з огляду на повну інформацію про поточне керування втікача, і базується на умові Л. С. Понтрягіна, яка відображає перевагу переслідувача над втікачем у ресурсах керування.

Умова 8.1. (Умова Понтрягіна [1])

$$W(t) = \pi e^{tA} U \dot{-} \pi e^{tA} V \neq \varnothing \quad \forall t \geq 0.$$

Коли ж ця умова не виконується на певному проміжку часу, то нижче пропонується її модифікація, яка передбачає, що переслідувач у процесі гри будує своє керування з огляду не на поточне керування втікача, а на його керування у минулому, так ніби інформація про керування втікача надходить до нього зі змінним запізненням у часі. Перед тим, як сформулювати модифіковану умову Л. С. Понтрягіна, введемо поняття функції розтягування часу.

Визначення 8.1. Функцією розтягування часу називатимемо невід'ємну монотонно зростаючу кусково-неперервну функцію $I(t)$, $t \in [0,+\infty)$, $I(0)=0$, $I(t) > t$, $t > 0$, яка може мати зліченну кількість розривів першого роду, абсолютно-неперервну на інтервалах своєї неперервності та таку, що $\sup\limits_{t \in [0,+\infty)\setminus\Delta} \dot{I}(t) < +\infty$, де Δ — множина точок розриву та недиференційованості функції $I(t)$.

Умова 8.2. Існує абсолютно неперервна функція розтягування часу $I(t)$, така, що багатозначне відображення

$$W_1(t) = \pi e^{tA} U \dot{-} \dot{I}(t) \pi e^{I(t)A} V,$$

має непорожні образи при t, $t \in [0,+\infty)$.

У подальшому буде використано поняття інтеграла Ауманна від багатозначного відображення [13]. Нижче наведено його визначення.

Нехай $F(t)$ це багатозначне відображення, $F:[t_0,T]\to P(R^n)$, де $P(R^n)$ це сукупність усіх замкнених множин з простору R^n. Об'єднання, взяте по всіх вимірних вибірках $f(t)$, $f(t)\in F(t)$:

$$\bigcup_{f(\cdot)\in F(\cdot)}\int_{t_0}^{T} f(t)\,dt,$$

називають інтегралом Ауманна від багатозначного відображення $F(t)$ і позначають $\int_{t_0}^{T} F(t)\,dt$.

Теорема 8.2. Нехай для диференціальної гри (8.1) з термінальною множиною M_0 — лінійним підпростором, виконана умова 8.2 та для заданого початкового стану z_0 існує скінченний момент часу

$$t_1 = t_1(z_0) = \min\left\{ t\ge 0 : -\pi\left(e^{I(t)A}z_0 + \int_0^{I(t)-t} e^{(I(t)-\theta)A}U\,d\theta \right) \cap \int_0^{t} W_1(\theta)\,d\theta \neq \varnothing \right\}. \quad (8.8)$$

Тоді з початкового стану z_0 гра може бути завершена в момент часу $I(t_1)$ за будь-яких допустимих керуваннях втікача.

Доведення. З огляду на непорожність перетину у визначенні часу t_1 (8.8) та непорожності образів багатозначного відображення $W_1(\theta)$ (умова 8.2) існують допустиме керування $u^{\tau_0}(\theta)$, $\theta\in[0,\tau_0)$, $\tau_0 = I(t_1)-t_1$ та вимірний селектор $\omega_1(\theta)$, $\omega_1(\theta)\in W_1(\theta)$, $0\le\theta\le t_1$ багатозначного відображення $W_1(\theta)$, такі, що виконується рівність

$$-\pi\left(e^{I(t_1)A}z_0 + \int_0^{I(t_1)-t_1} e^{(I(t_1)-\theta)A}u^{\tau_0}(\theta)\,d\theta \right) = \int_0^{t_1}\omega_1(\theta)\,d\theta. \quad (8.9)$$

Поділимо інтервал часу $[0,I(t_1)]$ на дві частини: півінтервал $[0,\tau_0)$, де $\tau_0 = I(t_1)-t_1$, та відрізок часу $[\tau_0,I(t_1)]$. Пропонується такий спосіб керування переслідувача на проміжках часу $[0,\tau_0)$ та $[\tau_0,I(t_1)]$. На півінтервалі часу

$[0,\tau_0)$ керування переслідувача кладемо рівним $u^{\tau_0}(\theta)$, $\theta\in[0,\tau_0)$, а на відрізку часу $[\tau_0,\tau_0+t_1]$ керування переслідувача будується у вигляді вимірного розв'язку такого рівняння:

$$\pi e^{(t_1-\theta)A}u(\tau_0+\theta)=$$

$$=\dot{I}(t_1-\theta)\pi e^{I(t_1-\theta)A}v\big(I(t_1)-I(t_1-\theta)\big)+\omega_1(t_1-\theta),\ \theta\in[0,t_1], \qquad (8.10)$$

існування якого забезпечує теорема Філіпова – Кастена про вимірний вибір [14].

Отже, починаючи з моменту часу τ_0, тобто в кожний поточний момент часу $\tau_0+\theta$, $0\le\theta\le t_1$, переслідувач будує своє керування з огляду на керування втікача в момент часу $I(t_1)-I(t_1-\theta)$. Оскільки $I(t_1)=\tau_0+t_1$, $\tau_0=I(t_1)-t_1$, то момент часу $I(t_1)-I(t_1-\theta)$ може бути представлений у вигляді:

$$I(t_1)-I(t_1-\theta)=\tau_0+t_1-I(t_1-\theta)=\tau_0+\theta-\big(I(t_1-\theta)-(t_1-\theta)\big).$$

Звідси з природних міркувань доходимо висновку, що, починаючи з моменту часу τ_0, тобто в кожний поточний момент часу $\tau_0+\theta$, $0\le\theta\le t_1$, переслідувач обирає своє керування з керування втікача в минулому, так ніби інформація про поточне керування втікача приходить до нього із запізненням у часі $\tau(\tau_0+\theta)$, $\tau(\tau_0+\theta)=I(t_1-\theta)-(t_1-\theta)$.

Таким чином, при описаному способі керування переслідувача на проміжку часу $[\tau_0,\tau_0+t_1]$ гра зближення (8.7) з термінальною множиною M_0 розвивається як гра із запізненням інформації $\tau(\tau_0+t)$,

$$\tau(\tau_0+t)=I(t_1-t)-(t_1-t),\ t\in[0,t_1], \qquad (8.11)$$

За теоремою 8.1, з урахуванням зауваження 1, ця гра із запізненням інформації може бути завершена в момент часу $I(t_1)=\tau_0+t_1$ тоді і тільки тоді, коли за умовою непорожності множини $M_0\dot{-}V\big(\tau(I(t_1))\big)$, де згідно з (8.2)

$$V\big(\tau(I(t_1))\big)=\left\{x: x=\int_0^{\tau(I(t_1))}e^{\theta A}v(\theta)d\theta,\ v(\theta)\in\Omega_V\right\},$$

у цей самий момент часу $I(t_1)$ може бути завершена еквівалентна до неї гра зближення з повною інформацією, динаміка якої описується системою (8.4)

$$\dot{\tilde{z}}(\tau_0+t)=A\tilde{z}(\tau_0+t)+u(\tau_0+t)-$$
$$-\left(1-\dot{\tau}(\tau_0+t)\right)e^{\tau(\tau_0+t)A}v\left(\tau_0+t-\tau(\tau_0+t)\right),\ t\in[\tau_0,\tau_0+t_1],$$

з термінальною множиною, $M(t_1)$, $M(t_1)=M_0\dot{-}V\left(\tau\left(I(t_1)\right)\right)$.

Після підстановки конкретної функції запізнення інформації $\tau(\tau_0+t)$ (8.11) це рівняння набуває вигляду

$$\dot{\tilde{z}}(\tau_0+t)=A\tilde{z}(\tau_0+t)+u(\tau_0+t)-\dot{I}(t_1-t)e^{\left[I(t_1-t)-(t_1-t)\right]A}v\left(\tau_0+t_1-I(t_1-t)\right),$$

$$\tilde{z}(\tau_0)=e^{\tau_0 A}z_0+\int_0^{\tau_0}e^{(\tau_0-\theta)A}u^{\tau_0}(\theta)d\theta. \qquad (8.12)$$

З огляду на формулу (8.11) $\tau(\tau_0+t_1)=0$, звідки випливає, що, оскільки $V\left(I(t_1)\right)=\{0\}$, де $\{0\}$ — n-мірний нульовий вектор, то $M(t_1)=M_0$.

Таким чином, з огляду на спеціальний вигляд функції запізнення інформації (8.11), при переході від гри (8.7) із запізненням інформації (8.11) до еквівалентної до неї гри (8.12) з повною інформацією термінальною множиною залишається підпростір M_0 (завдяки тому, що в кінцевий момент гри запізнення зникає).

Представимо траєкторію системи (8.12) в момент $I(t_1)$ у вигляді формули Коші з подальшим урахуванням формули (8.10), за якою керування переслідувача будується на відрізку часу $[\tau_0,I(t_1)]$. Отримаємо низку рівностей:

$$\pi z\left(I(t_1)\right)=\pi\left(e^{I(t_1)A}z_0+\int_0^{\tau_0}e^{(I(t_1)-\theta)A}u^{\tau_0}(\theta)d\theta\right)+$$

$$+\int_0^{t_1}\left(\pi e^{(t_1-\theta)A}u(\tau_0+\theta)-\dot{I}(t_1-\theta)\pi e^{(I(t_1-\theta))A}v\left(\tau_0+t_1-I(t_1-\theta)\right)\right)d\theta=$$

$$=\pi\left(e^{I(t_1)A}z_0+\int_0^{\tau_0}e^{(I(t_1)-\theta)A}u^{\tau_0}(\theta)d\theta\right)+\int_0^{t_1}\omega_1(t_1-\theta)d\theta.$$

З огляду на співвідношення (8.9) остання рівність означає, що $\pi z\left(I\left(t_1\right)\right)=0$. Отже, доведено, що переслідувач, обираючи керування вищеописаним способом, виведе траєкторію системи (8.12) на термінальну множину M_0 за будь-якої протидії втікача.

Повернемося від гри (8.12) з повною інформацією до еквівалентної до неї гри (8.7) із запізненням інформації (8.11), яка починається з початкової позиції $\left(z_0, u^0(\theta)\right)$ і розвивається на відрізку часу $\left[\tau_0, I\left(t_1\right)\right]$. За теоремою 8.1, переслідувач може завершити цю гру в момент часу $I\left(t_1\right)$. Оскільки перехід до гри із запізненням інформації носить штучний (допоміжний) характер і лише відбиває спосіб керування переслідувача з керування втікача в минулому, останнє означає, що переслідувач, обираючи керування з керування втікача в минулому, виведе траєкторію вихідної гри (8.7) з повною інформацією з точки z_0 на термінальну множину M у момент часу $I\left(t_1\right)$ за будь-яких допустимих керувань втікача. Теорему доведено.

8.4. М'яке зближення керованих систем другого порядку, що описують ньютонівську динаміку за наявності тертя

Розглянемо модельну гру (контрольний приклад Л. С. Понтрягіна) про м'яку зустріч об'єктів, рух яких описується диференціальними рівняннями другого порядку:

$$\ddot{x}+\alpha\dot{x}=\rho u, \quad \ddot{y}+\beta\dot{y}=\sigma v, \tag{8.13}$$

де $x, y \in R^n$, $\|u\| \le 1$, $\|v\| \le 1$. Тут x, y — геометричні положення відповідно переслідувача та втікача, u, v — їхнє керування, α і $\beta-$ коефіцієнти тертя, ρ і σ — силові коефіцієнти, $\alpha, \beta, \rho, \sigma > 0$.

Задані початкові стани та швидкості об'єктів:

$$x(0)=x_0, \dot{x}(0)=\dot{x}_0, y(0)=y_0, \dot{y}(0)=\dot{y}_0.$$

Мета переслідувача — за допомогою вибору свого керування за будь-яких допустимих керуваннях супротивника досягти в деякий скінчений момент часу t одночасного збігу геометричних координат та швидкостей об'єктів: $x(t) = y(t),\ \dot{x}(t) = \dot{y}(t)$.

За допомогою стандартної заміни змінних:

$$x_1 = x,\quad x_2 = \dot{x},\quad y_1 = y,\quad y_2 = \dot{y}$$

перейдемо від рівнянь другого порядку (8.13) до системи рівнянь першого порядку відносно змінної $z = (x_1, x_2, y_1, y_2),\ z \in R^{4n}$,

$$\begin{aligned}
&\dot{x}_1 = x_2,\\
&\dot{x}_2 = -\alpha x_2 + \rho u,\\
&\dot{y}_1 = y_2,\\
&\dot{y}_2 = -\beta y_2 + \sigma v,
\end{aligned}$$

з початковою умовою

$$z(0) = z_0 = (x_0, \dot{x}_0, y_0, \dot{y}_0).$$

Фундаментальна матриця цієї системи є такою:

$$e^{At} = \begin{pmatrix}
E & \dfrac{1-e^{-\alpha t}}{\alpha}E & O & O\\
O & e^{-\alpha t}E & E & O\\
O & O & E & \dfrac{1-e^{-\beta t}}{\alpha}E\\
O & O & O & e^{-\beta t}E
\end{pmatrix},$$

де E та O — квадратні одинична та нульова матриці порядку n.

Термінальна множина гри є лінійним підпростором з простору R^{4n}:

$$M = \left\{(x_1, x_2, y_1, y_2),\quad x_1, x_2, y_1, y_2 \in R^n :\ x_1 = y_1,\ x_2 = y_2\right\}.$$

Тоді м'яка зустріч у деякий момент часу t, $t > 0$, означає, що $\pi z(t) = 0$. Ортогональним доповненням до M в R^{4n} є лінійний підпростір

$$L = \left\{(x_1, x_2, y_1, y_2),\quad x_1, x_2, y_1, y_2 \in R^n :\ x_1 = -y_1,\ x_2 = -y_2\right\},$$

оператор ортогонального проектування з R^{4n} на L задається матрицею

$$\pi = \begin{pmatrix} E & O & -E & 0 \\ O & E & O & -E \end{pmatrix},$$

а множини керувань U та V мають вигляд

$$U = \begin{pmatrix} O \\ \rho S \\ O \\ O \end{pmatrix},\ V = \begin{pmatrix} O \\ O \\ O \\ \sigma S \end{pmatrix},$$

де $S, S \in R^n$, — одинична куля з центром у нулі. Неважко бачити, що множини $\pi e^{At} U, \pi e^{At} V$ складаються з векторів, відповідно,

$$\left(\rho \frac{1-e^{-\alpha t}}{\alpha} u, \rho e^{-\alpha t} u \right)^T,\ \left(\sigma \frac{1-e^{-\beta t}}{\beta} v, \sigma e^{-\beta t} v \right)^T,\ \|u\| \le 1,\ \|v\| \le 1.$$

Багатозначне відображення $W(t)$, яке фігурує в умові 8.1, має вигляд:

$$W(t) = \bigcap_{\|v\| \le 1} \bigcup_{\|u\| \le 1} \begin{pmatrix} \rho \frac{1-e^{-\alpha t}}{\alpha} u - \sigma \frac{1-e^{-\beta t}}{\beta} v \\ \rho e^{-\alpha t} u - \sigma e^{-\beta t} v \end{pmatrix}.$$

Аналіз умови Понтрягіна для цієї задачі, проведений М. С. Нікольським у [4] показав, що множина $W(t)$ є непорожньою в єдиній точці $t = 0$ при $\rho \ge \sigma$, а при $\rho < \sigma$ не існує інтервалу часу з лівим кінцем у нулі, на якому $W(t) \ne \varnothing$. Отже, умова 8.1, а саме $W(t) \ne \varnothing\ \ \forall t \ge 0$, тут не виконується.

Запишемо умову 8.2 та знайдемо функцію $I(t)$, що задовольняє цю умову. У прикладі, що розглядається, багатозначне відображення $W_1(t)$ має вигляд:

$$W_1(t) = \bigcap_{\|v\| \le 1} \bigcup_{\|u\| \le 1} \begin{pmatrix} \rho \frac{1-e^{-\alpha t}}{\alpha} u - \sigma \dot{I}(t) \frac{1-e^{-\beta I(t)}}{\beta} v \\ \rho e^{-\alpha t} u - \sigma \dot{I}(t) e^{-\beta I(t)} v \end{pmatrix}. \qquad (8.14)$$

Умова 8.2, а саме непорожність множини $W_1(t)$ при всіх $t \ge 0$ має місце, якщо існує пара n-мірних векторів $d_1(t), d_2(t)$, таких, що для кожного вектора

v, $v \in R^n, \|v\| \le 1$, знайдеться вектор $u, u \in R^n, \|u\| \le 1$, для яких одночасно виконуються такі рівності:

$$\begin{aligned} &\sigma\, \dot{I}(t)\frac{1-e^{-\beta I(t)}}{\beta}v + d_1(t) = \rho\,\frac{1-e^{-\alpha t}}{\alpha}u, \\ &\sigma\, \dot{I}(t)e^{-\beta I(t)}v + d_2(t) = \rho\, e^{-\alpha t}u. \end{aligned} \tag{8.15}$$

Зокрема рівності повинні виконуватись при $v = 0$, тому з (8.15) отримаємо

$$d_1(t) = \rho\,\frac{1-e^{-\alpha t}}{\alpha}u_0,\ d_2(t) = \rho\, e^{-\alpha t}u_0,$$

де u_0 – керування, що відповідає $v = 0$. Підставивши $d_1(t)$, $d_2(t)$ в систему (8.15), отримаємо рівності

$$\sigma\dot{I}(t)\frac{1-e^{-\beta I(t)}}{\beta}v = \rho\frac{1-e^{-\alpha t}}{\alpha}(u-u_0),$$

$$\sigma\dot{I}(t)e^{-\beta I(t)}v = \rho e^{-\alpha t}(u-u_0),$$

з яких випливає співвідношення

$$\frac{1-e^{-\beta I(t)}}{\beta} = \frac{1-e^{-\alpha t}}{\alpha}e^{\alpha t-\beta I(t)}. \tag{8.16}$$

Звідси отримаємо явний вигляд функції $I(t)$, $t \in [0,+\infty)$:

$$I(t) = \frac{1}{\beta}\ln\left[\beta\Big/\alpha\left(e^{\alpha t}-1\right)+1\right]. \tag{8.17}$$

Бачимо, що $I(t)$, $t \ge 0$, — невід'ємна, неперервно-диференційована функція, $I(0) = 0$. Знайдемо похідну функції $I(t)$.

$$\dot{I}(t) = e^{\alpha t}\left(\frac{\beta}{\alpha}\left(e^{\alpha t}-1\right)+1\right)^{-1}. \tag{8.18}$$

Припустимо, що $\alpha > \beta$. Легко впевнитись, що $\dot{I}(t)$ монотонно зростає, $\dot{I}(0) = 1$, $1 < \dot{I}(t) < \alpha/\beta$, $t > 0$, оскільки

$$\sup_{t\in[0,+\infty)} \dot{I}(t) = \frac{\alpha}{\beta}. \tag{8.19}$$

Бачимо, що $I(t)$ — монотонно зростаюча функція, $I(t) > t$ при $t > 0$.

Отже, за умови $\alpha > \beta$, функція $I(t)$ задовольняє визначенню 8.1, тобто є функцією розтягування часу.

З формули (8.17) випливає, що $e^{\beta I(t)} = \beta/\alpha\left(e^{\alpha t} - 1\right) + 1$, а, врахувавши вираз (8.18) для похідної, отримаємо $\dot{I}(t) = e^{\alpha t}\, e^{-\beta I(t)}$. Тоді формула (8.16) дає

$$\frac{1 - e^{-\beta I(t)}}{\beta} = \dot{I}(t)\frac{1 - e^{-\alpha t}}{\alpha}. \tag{8.20}$$

Очевидно, що має місце рівність

$$e^{-\beta I(t)} = \dot{I}(t) e^{-\alpha t}. \tag{8.21}$$

Бачимо, що для гри, яка розглядається, має місце співвідношення

$$\pi e^{tA} U = k(t)\pi e^{I(t)A} V, \tag{8.22}$$

де $k(t) = \dot{I}(t)$.

З огляду на формули (8.20), (8.21) множина $W_1(t)$ (8.14) набуває вигляду

$$W_1(t) = \bigcap_{\|v\| \le 1} \bigcup_{\|u\| \le 1} \begin{pmatrix} \rho \dfrac{1 - e^{-\alpha t}}{\alpha} u - \sigma\left(\dot{I}(t)\right)^2 \dfrac{1 - e^{-\alpha t}}{\beta} v \\ \rho e^{-\alpha t} u - \sigma\left(\dot{I}(t)\right)^2 e^{-\alpha t} v \end{pmatrix} =$$

$$= \bigcup_{s \in R^n, \|s\| \le 1} \rho \begin{pmatrix} \dfrac{1 - e^{-\alpha t}}{\alpha} s \\ e^{-\alpha t} s \end{pmatrix} \stackrel{*}{-} \bigcup_{s \in R^n, \|s\| \le 1} \sigma\left(\dot{I}(t)\right)^2 \begin{pmatrix} \dfrac{1 - e^{-\alpha t}}{\alpha} s \\ e^{-\alpha t} s \end{pmatrix} =$$

$$= \left(\rho - \sigma\left(\dot{I}(t)\right)^2\right) \overline{S}(t),$$

де множина $\overline{S}(t)$:

$$\overline{S}(t) = \bigcup_{s \in R^n, \|s\| \le 1} \begin{pmatrix} \dfrac{1 - e^{-\alpha t}}{\alpha} s \\ e^{-\alpha t} s \end{pmatrix}.$$

Якщо $\sup\limits_{0 \le \theta < t < +\infty} \left(\dot{I}(t)\right)^2 \le \rho/\sigma$, то

$$\rho - \sigma\left(\dot{I}(t)\right)^2 > \rho - \sigma\alpha^2/\beta^2 \geq 0$$

і множина $W_1(t)$ є непорожньою при всіх $t \geq 0$. З огляду на оцінку (8.19) це має місце, якщо $\rho \geq \sigma\dfrac{\alpha^2}{\beta^2}$. Отже, виконання умови 8.2 у цій модельній грі забезпечується такими умовами на параметри гри:

$$\alpha > \beta,\ \frac{\rho}{\alpha^2} \geq \frac{\sigma}{\beta^2}. \tag{8.23}$$

Ці умови мають простий механічний зміст. Перша з них означає, що коефіцієнт тертя у переслідувача більше, ніж у втікача. З метою роз'яснення другої умови зазначимо, що для довільного додатного числа ε існує час $t(\varepsilon) > 0,$ такий, що при $t > t(\varepsilon)$ $\left\|\dot{x}(t)\right\| \leq \dfrac{\rho}{\alpha} + \varepsilon,\ \left\|\dot{y}(t)\right\| \leq \dfrac{\sigma}{\beta} + \varepsilon.$ Тоді друга умова в (8.23) свідчить, що відношення граничної швидкості до коефіцієнта тертя у переслідувача більше, ніж у втікача.

Нехай параметри систем (8.13) задовольняють нерівностям (8.23). Доведемо, що для довільних початкових положень та швидкостей переслідувача та втікача $(x_0, \dot{x}_0)$, $(y_0, \dot{y}_0)$ існує такий скінчений момент часу $t_1, t_1 > 0,$ для якого має місце включення (8.8).

Для цього пропонується такий метод побудови керування переслідувача. На початковому півінтервалі часу керування переслідувача покладемо тотожно рівним нулю: $u^0(\theta) = 0, \quad \theta \in \left[0, I(t_1) - t_1\right)$. Тоді з огляду на умову (8.9) теореми 8.2 залишилося довести існування скінченого моменту часу $t = t_1,\ 0 < t_1 < +\infty$, для якого виконується включення

$$-\pi e^{I(t)A} z_0 \in \int_0^t W_1(\theta)\, d\theta. \tag{8.24}$$

У цьому прикладі права частина включення (8.24), після врахування співвідношень (8.20)–(8.21), набуває вигляду

$$\int_0^t W_1(\theta)\,d\theta = \left(\rho - \sigma\left(\dot{I}(t)\right)^2\right)\overline{\overline{S}}(t),$$

де

$$\overline{\overline{S}}(t) = \bigcup_{s\in R^n,\,\|s\|\le 1}\begin{pmatrix} \int_0^t \dfrac{1-e^{-\alpha\theta}}{\alpha}d\theta\cdot s \\ \int_0^t e^{-\alpha\theta}d\theta\cdot s \end{pmatrix},$$

і з огляду на оцінку (8.19) та умову на параметри гри (8.23) містить у собі множину $\left(\rho - \sigma\alpha^2/\beta^2\right)S(t)$, де

$$S(t) = \bigcup_{s\in R^n,\,\|s\|\le 1}\begin{pmatrix} \left(t - \dfrac{1-e^{-\alpha t}}{\alpha}\right)s \\ \dfrac{1-e^{-\alpha t}}{\alpha}s \end{pmatrix}.$$

Вектор, що стоїть у лівій частині включення (8.24), має вигляд

$$-\pi e^{I(t)A}z_0 = \begin{pmatrix} y_0 - x_0 + \dfrac{1-e^{-\beta I(t)}}{\beta}\dot{y}_0 - \dfrac{1-e^{-\alpha I(t)}}{\alpha}\dot{x}_0 \\ e^{-\beta I(t)}\dot{y}_0 - e^{-\alpha I(t)}\dot{x}_0 \end{pmatrix}$$

і при $t\to+\infty$, прямує до вектора

$$\begin{pmatrix} y_0 - x_0 + \dfrac{\dot{y}_0}{\beta} - \dfrac{\dot{x}_0}{\alpha} \\ 0 \end{pmatrix},$$

тому в деякий момент часу t_1, $t_1<\infty$ він буде поглинутий множиною $\left(\rho - \sigma\alpha^2/\beta^2\right)S(t)$.

Це означає, що в цей момент часу t_1 виповниться включення (8.24), тобто буде виконано припущення теореми 8.2. Скориставшись цією теоремою, доходимо висновку, що за будь-яких початкових положеннях та швидкостях переслідувача $(x_0, \dot{x}_0)$ та втікача $(y_0, \dot{y}_0)$ існує такий скінчений момент часу t_1,

що переслідувач, будуючи керування на відрізку $\left[I(t_1)-t_1, I(t_1)\right]$ з керування супротивника в минулому (8.10), здійснить з ним м'яку зустріч у момент часу $I(t_1)$.

Умова взяття сліду. Окремо дослідимо спеціальний випадок непорожності перетину в (8.8), коли вказаний перетин множин містить початок координат, тобто

$$\{0\} \in \pi\left(e^{I(t)A}z_0 + \int_0^{I(t)-t} e^{(I(t)-\theta)A}Ud\theta\right) \cap \int_0^t W_1(\theta)d\theta, \tag{8.25}$$

де $\{0\}$ – нульовий вектор з R^{2n}. З формули (8.25) випливають два включення:

$$\{0\} \in \pi\left(e^{I(t)A}z_0 + \int_0^{I(t)-t} e^{(I(t)-\theta)A}Ud\theta\right), \tag{8.26}$$

$$\{0\} \in \int_0^t \left(\rho\pi e^{\theta A}U \stackrel{*}{-} \dot{I}(t)\pi e^{I(\theta)A}V\right)d\theta. \tag{8.27}$$

Для конкретної гри, що розглядається, включення (8.26) означає, що існує допустиме керування $u^{\tau_0}(t), t \in \left[0, I(t_1)-t_1\right)$, таке, що виконуються рівності

$$-(x_0-y_0)-\left(\frac{1-e^{-\alpha I(t_1)}}{\alpha}\dot{x}_0 - \frac{1-e^{-\beta I(t_1)}}{\beta}\dot{y}_0\right) = \int_0^{I(t_1)-t_1} \frac{1-e^{-\alpha(I(t_1)-\theta)}}{\alpha}u^{\tau_0}(\theta)d\theta,$$

$$-\left(e^{-\alpha I(t_1)}\dot{x}_0 - e^{-\beta I(t_1)}\dot{y}_0\right) = \int_0^{I(t_1)-t_1} e^{-\alpha(I(t_1)-\theta)}u^{\tau_0}(\theta)d\theta.$$

Врахувавши співвідношення (8.20), (8.21), звідси маємо

$$\begin{aligned} -(x_0-y_0)-\left(\frac{1-e^{-\alpha I(t_1)}}{\alpha}\dot{x}_0 - \dot{I}(t_1)\frac{1-e^{-\alpha t_1}}{\alpha}\dot{y}_0\right) &= \int_0^{I(t_1)-t_1} \frac{1-e^{-\alpha(I(t_1)-\theta)}}{\alpha}u^{\tau_0}(\theta)d\theta, \\ -\left(e^{-\alpha I(t_1)}\dot{x}_0 - \dot{I}(t_1)e^{-\alpha t_1}\dot{y}_0\right) &= \int_0^{I(t_1)-t_1} e^{-\alpha(I(t_1)-\theta)}u^{\tau_0}(\theta)d\theta. \end{aligned} \tag{8.28}$$

Помноживши обидві частини другої рівності в (8.28) на $e^{\alpha t_1}$, отримаємо

$$e^{-\alpha(I(t_1)-t_1)}\dot{x}_0 + \int_0^{I(t_1)-t_1} \frac{1-e^{-\alpha(I(t_1)-t_1-\theta)}}{\alpha}u^0(\theta)d\theta = e^{\alpha t_1-\beta I(t_1)}\dot{y}_0.$$

Оскільки $e^{\alpha t_1-\beta I(t_1)}=\dot{I}(t_1)$ (8.21), звідси випливає співвідношення

$$x_2(\tau_0)=\dot{I}(t_1)\dot{y}_0, \tag{8.29}$$

де $\tau_0=I(t_1)-t_1$.

Далі у вираз для x_0, отриманий з першої рівності в (8.28):

$$x_0=y_0-\left(\frac{1-e^{-\alpha I(t_1)}}{\alpha}\dot{x}_0-\dot{I}(t_1)\frac{1-e^{-\alpha t_1}}{\alpha}\dot{y}_0\right)-\int_0^{I(t_1)-t_1}\frac{1-e^{-\alpha(I(t_1)-\theta)}}{\alpha}u^{\tau_0}(\theta)d\theta,$$

підставимо у формулу Коші, записану для $x_1(\tau_0)$:

$$x_1(\tau_0)=x_0+\frac{1-e^{-\alpha\tau_0}}{\alpha}\dot{x}_0+\int_0^{\tau_0}\frac{1-e^{-\alpha(\tau_0-\theta)}}{\alpha}u^{\tau_0}(\theta)d\theta.$$

Тоді отримаємо

$$x_1(\tau_0)=y_0-\frac{1-e^{-\alpha I(t_1)}}{\alpha}\dot{x}_0+\dot{I}(t_1)\frac{1-e^{-\alpha t_1}}{\beta}\dot{y}_0-\int_0^{\tau_0}\frac{1-e^{-\alpha(I(t_1)-\theta)}}{\alpha}u^{\tau_0}(\theta)d\theta+$$

$$+\frac{1-e^{-\alpha\tau_0}}{\alpha}\dot{x}_0+\int_0^{\tau_0}\frac{1-e^{-\alpha(\tau_0-\theta)}}{\alpha}u^{\tau_0}(\theta)d\theta.$$

Після перегрупування доданків у правій частині формули та простих перетворень, отримаємо

$$x_1(\tau_0)=y_0+\frac{1}{\alpha}(\dot{x}_0-\dot{x}_0)+\frac{1}{\alpha}\left(e^{-\alpha I(t_1)}\dot{x}_0-e^{-\alpha\tau_0}\dot{x}_0\right)+\left(\dot{I}(t_1)e^{-\alpha t_1}\dot{y}_0-\dot{I}(t_1)e^{-\alpha t_1}\dot{y}_0\right)+$$

$$+\left(\int_0^{\tau_0}\frac{1-e^{-\alpha(I(t_1)-\theta)}}{\alpha}u^{\tau_0}(\theta)d\theta-\int_0^{\tau_0}\frac{1-e^{-\alpha(I(t_1)-\theta)}}{\alpha}u^{\tau_0}(\theta)d\theta\right)=$$

$$=y_0+\frac{1}{\alpha}\left(e^{-\alpha I(t_1)}\dot{x}_0+\int_0^{\tau_0}\frac{1-e^{-\alpha(I(t_1)-\theta)}}{\alpha}u^{\tau_0}(\theta)d\theta-e^{-\alpha\tau_0}\dot{x}_0-\int_0^{\tau_0}\frac{1-e^{-\alpha(\tau_0-\theta)}}{\alpha}u^{\tau_0}(\theta)d\theta\right)=$$

$$=y_0+\frac{1}{\alpha}\left(x_2(I(t_1))-\int_{I(t_1)-t_1}^{I(t_1)}e^{-\alpha(I(t_1)-\theta)}u^0(\theta)d\theta-x_2(\tau_0)\right)=y_0+$$

$$+\frac{1}{\alpha}\left(x_2(I(t_1))-\left(x_2(\tau_0)+\int_{I(t_1)-t_1}^{I(t_1)}e^{-\alpha(I(t_1)-\theta)}u^0(\theta)d\theta\right)\right)=y_0+\frac{1}{\alpha}\left[x_2(I(t_1))-x_2(I(t_1))\right].$$

Звідси маємо

$$x_1(\tau_0)=y_0. \tag{8.30}$$

Отже, показано, що включення (8.25) еквівалентне умовам (8.29), (8.30).

Включення (8.27) тут має вигляд:

$$\{0\} \in \int_0^{t_1} W_1(\theta)\,d\theta = \bigcup_{s\in R^n,\,\|s\|\le 1} \begin{pmatrix} \int_0^{t_1} \left(\rho - \sigma\left(\dot{I}(\theta)\right)\right)^2 \dfrac{1-e^{-\alpha\theta}}{\beta} d\theta \cdot s \\ \int_0^{t_1} \left(\rho - \sigma\left(\dot{I}(\theta)\right)\right)^2 e^{-\alpha\theta} d\theta \cdot s \end{pmatrix} \tag{8.31}$$

де $\{0\}$ — нульовий вектор з простору R^{2n} і, з огляду на співвідношення (8.19)–(8.21), його виконання забезпечується умовами (8.23).

Уявимо таку ситуацію. У деякий момент $\tau_0 = I(t_1) - t_1$, де функція $I(t)$ визначена формулою (8.18), а $t_1 > 0$, виявилось, що переслідувач знаходиться у положенні, в якому втікач був у момент $t = 0$, тобто має місце рівність (8.30), а його швидкість пов'язана з початковою швидкістю втікача в момент $t = 0$ співвідношенням (8.29), тобто збігається з нею за напрямом, але перевищує за нормою. Назвемо таку ситуацію умовою «взяття сліду». Тоді, як випливає з включення (8.27) та формул (8.23), переслідувач, застосовуючи керування за формулою

$$u(\tau_0 + t) = \sigma \Big/ \rho\left(\dot{I}(t_1 - t)\right)^2 v\left(\tau_0 + t_1 - I(t_1 - t)\right), \quad 0 \le t \le t_1, \tag{8.32}$$

досягне м'якої зустрічі з втікачем точно в момент часу $I(t_1) = \tau_0 + t_1$. Це випливає з теореми 8.1, оскільки з огляду на вибір керування (8.32) та за умови взяття сліду (8.29), (8.30), виконуються включення (8.26)–(8.27), а, отже, і включення (8.25) (умова теореми 8.2).

Покажемо, що в процесі гри, вибираючи керування за формулою (8.32), переслідувач рухається уздовж геометричного сліду втікача аж до моменту м'якої зустрічі, тобто виконуються співвідношення

$$x(\tau_0 + t) = y\left(\tau_0 + t - \tau(\tau_0 + t)\right), \quad t \in [0, t_1], \tag{8.33}$$

де $\tau(\tau_0 + t)$ — запізнення у часі, визначене формулами (8.6), (8.17), а його швидкість пов'язана зі швидкістю втікача таким чином:

$$x_2(\tau_0 + t) = \dot{I}(t_1 - t)\, y_2\left(\tau_0 + t - \tau(\tau_0 + t)\right). \tag{8.34}$$

Спочатку доведемо співвідношення (8.34). Оскільки $\tau_0 = I(t_1) - t_1$, $\tau(\tau_0 + t) = I(t_1 - t) - (t_1 - t)$ 11), то $\tau_0 + t - \tau(\tau_0 + t) = I(t_1) - I(t_1 - t)$.

Зауважимо, що з огляду на співвідношення $e^{-\beta I(t)} = \dot{I}(t)e^{-\alpha t}$ (8.21), маємо

$$e^{-\beta(I(t_1)-I(t_1-t))} = \frac{e^{-\beta I(t_1)}}{e^{\beta I(t_1-t)}} = \frac{\dot{I}(t_1)e^{-\alpha t_1}}{\dot{I}(t_1-t)e^{-\alpha(t_1-t)}} = \frac{\dot{I}(t_1)}{\dot{I}(t_1-t)}e^{-\alpha t},$$

звідки випливають формули:

$$e^{-\alpha t} = \dot{I}(t_1)^{-1}\dot{I}(t_1-t)e^{-\beta(I(t_1)-I(t_1-t))},\ 0 \le t \le t_1,$$

$$e^{-\alpha(t-\theta)} = \dot{I}(t_1)^{-1}\dot{I}(t_1-t)e^{-\beta(I(t_1)-I(t_1-t))}e^{\alpha\theta} = \dot{I}(t_1)^{-1}\dot{I}(t_1-t)e^{-\beta I(t_1)}e^{-\beta I(t_1-t)} =$$

$$= \dot{I}(t_1-t)\left(\dot{I}(t_1-\theta)\right)^{-1}e^{\beta(I(t_1)-I(t_1-\theta))},\ 0 \le \theta \le t \le t_1.$$

Підставимо у формулу Коші для $x_2(\tau_0 + t)$ співвідношення (8.29) і формулу для керування переслідувача (8.9) та врахуємо отримані вище вирази.

$$x_2(\tau_0 + t) = e^{-\alpha t}x_2(\tau_0) + \rho\int_0^t e^{-\alpha(t-\theta)}u(\tau_0 + \theta)d\theta =$$

$$= e^{-\alpha t}\dot{I}(t_1)\dot{y}_0 + \sigma\int_0^t \mathrm{e}^{-\alpha(\mathrm{t}-\theta)}\left(\dot{I}(t_1-\theta)\right)^2 v(\tau_0 + t_1 - I(t_1-\theta))d\theta =$$

$$= \dot{I}(t_1-t)e^{-\beta(I(t_1)-I(t_1-t))}\dot{y}_0 +$$

$$+\sigma\int_0^t \dot{I}(t_1-t)\left(\dot{I}(t_1-\theta)\right)^{-1}\left(\dot{I}(t_1-\theta)\right)^2 e^{\beta(I(t_1-t)-I(t_1-\theta))}v(\tau_0 + t_1 - I(t_1-\theta))d\theta.$$

З огляду на те, що

$$I(t_1-t) - I(t_1-\theta) = -\left(\left(I(t_1) - I(t_1-t)\right) - \left(I(t_1) - I(t_1-\theta)\right)\right),$$

перепишемо формулу для $x_2(\tau_0 + t)$ у вигляді

$$x_2(\tau_0 + t) =$$

$$= \dot{I}(t_1-t)\left(e^{-\beta(I(t_1)-I(t_1-t))}\dot{y}_0 + \sigma\int_0^t \dot{I}(t_1-\theta)e^{-\beta\left((I(t_1)-I(t_1-t))-(I(t_1)-I(t_1-\theta))\right)}v\left(I(t_1) - I(t_1-\theta)\right)d\theta\right).$$

Зробимо заміну змінної під знаком інтеграла $\theta_1 = \left(I(t_1) - I(t_1-\theta)\right)$. Отримаємо

$$x_2(\tau_0+t)=\dot I(t_1-t)e^{-\beta(I(t_1)-I(t_1-t))}\dot y_0+\int_0^{I(t_1)-I(t_1-t)}e^{-\beta(I(t_1)-I(t_1-t)-\theta_1)}v(\theta_1)d\theta_1=$$

$$=\dot I(t_1-t)y\big(I(t_1)-I(t_1-t)\big)=\dot I(t_1-t)y\big(\tau_0+t-\tau(\tau_0+t)\big).$$

Отже, рівність (8.34) доведена.

За формулою Коші,

$$x_1(\tau_0+t)=x_1(\tau_0)+\int_0^t x_2(\tau_0+\theta)d\theta,$$

звідки, з огляду на умову (8.31) та співвідношення (8.34), маємо

$$x_1(\tau_0+t)=y_1(0)+\int_0^t\dot I(t_1-\theta)y_2\big(I(t_1)-I(t_1-\theta)\big)d\theta.$$

Зробимо заміну змінної під знаком інтеграла $\theta_1=I(t_1)-I(t_1-\theta)$.

Отримаємо

$$x_1(\tau_0+t)=y_1(0)+\int_0^{I(t_1)-I(t_1-t)}y_2(\theta)d\theta=y_1\big(I(t_1)-I(t_1-t)\big)$$

$$=y_1\big(\tau_0+t-\tau(\tau_0+t)\big).$$

Отже, співвідношення (8.34) доведено.

6.5. М'яке зближення керованих систем другого порядку, що описують динаміку математичних маятників

Дослідимо задачу про м'яку зустріч двох конфліктно-керованих об'єктів, рух яких описується системами

$$\begin{aligned}&\ddot x+a^2x=\rho u,\ x\in R^n,\\&\ddot y+b^2y=\sigma v,\ y\in R^n,\end{aligned}\tag{8.35}$$

де x та y це геометричні координати гравців, u та v — їх керування. $\|u\|\le 1$, $\|v\|\le 1$, a^2 та b^2 — коефіцієнти жорсткості систем, ρ та σ — силові коефіцієнти, a,b,ρ,σ — додатні числа. Числа a і b це власні частоти колових коливань відповідних систем (8.35) [15], $a>b$.

Задані початкові положення та швидкості учасників

$$x(0)=x_0,\ \dot{x}(0)=\dot{x}_0,\ y(0)=y_0,\ \dot{y}(0)=\dot{y}_0. \tag{8.36}$$

Як і в попередньому прикладі, за допомогою заміни змінних

$$z_1=x,\quad z_2=\dot{x},\quad z_3=y,\quad z_4=\dot{y}$$

перейдемо від систем другого порядку (8.35) до системи першого порядку вигляду (8.7):

$$\begin{aligned}
&\dot{z}_1=z_2,\\
&\dot{z}_2=-a^2z_1+\rho u,\\
&\dot{z}_3=z_4,\\
&\dot{z}_4=-b^2z_3+\sigma v,
\end{aligned} \tag{8.37}$$

з початковими умовами: $z_1(0)=x_0,\ z_2(0)=\dot{x}_0,\ z_3(0)=y_0,\ z_4(0)=\dot{y}_0$.

Бачимо, що термінальна множина M_0 та оператор π є такими самим, як і в попередньому модельному прикладі:

$$M_0=\left\{(x_1,x_2,y_1,y_2),\ x_1,x_2,y_1,y_2\in R^n:\ x_1=y_1,\ x_2=y_2\right\},$$

$$\pi=\begin{pmatrix} E & 0 & -E & 0\\ 0 & E & 0 & -E\end{pmatrix},\ U=\begin{pmatrix}O & \rho S & O & O\end{pmatrix},\ U=\begin{pmatrix}O & O & O & \sigma S\end{pmatrix}.$$

М’яка зустріч об'єктів у деякий момент часу t, $t>0$, означає, що $\pi z(t)=0$.

Тут фундаментальна матриця об’єднаної системи є такою [15]:

$$e^{At}=\begin{pmatrix}
\cos at\cdot E & \frac{1}{a}\sin at\cdot E & O & O\\
-a\sin at\cdot E & \cos at\cdot E & O & O\\
O & O & \cos bt\cdot E & \frac{1}{b}\sin bt\cdot E\\
O & O & -b\sin bt\cdot E & \cos bt\cdot E
\end{pmatrix},$$

де E та O — квадратні одинична та нульова матриці порядку n відповідно.

Умова Понтрягіна для гри, що розглядається, тут має вигляд:

$$\bigcap_{\|v\|\le 1}\bigcup_{\|u\|\le 1}\begin{pmatrix}\frac{\rho}{a}|\sin at|u-\frac{\sigma}{b}|\sin bt|v\\ \rho|\cos at|u-\sigma|\cos bt|v\end{pmatrix}\neq\varnothing \qquad \forall t\ge 0.$$

Для її виконання необхідно, щоб задовольнялась досить обтяжливі умови, а саме: $b=(2k-1)a$, де $k-$ натуральне число, та

$$\rho\frac{b}{a}\left|\frac{tgat}{tgbt}\right|\ge\sigma,\quad t\in\left[0,\frac{\pi}{2a}\right].$$

У протилежному випадку умова Понтрягіна може виконуватися лише періодично в часі [3].

Запишемо умову 8.2.

$$W_1(t)=\bigcap_{\|v\|\le 1}\bigcup_{u\le 1}\begin{pmatrix}\frac{\rho}{a}|\sin at|u-\frac{\sigma}{b}\dot I(t)|\sin bI(t)|v\\ \rho|\cos at|u-\sigma\dot I(t)|\cos bI(t)|v\end{pmatrix}\neq\varnothing\quad\forall t\ge 0. \qquad (8.38)$$

Ця умова виконується, якщо існує пара n-мірних векторів $d_1(t), d_2(t)$, таких, що для кожного вектора v, $v\in R^n, \|v\|\le 1$, знайдеться вектор $u, u\in R^n, \|u\|\le 1$, для яких одночасно виконуються такі рівності:

$$\frac{\sigma}{\beta}\dot I(t)|\sin bI(t)|v+d_1(t)=\frac{\rho}{\alpha}|\sin at|u$$

$$\sigma\dot I(t)|\cos bI(t)|v+d_2(t)=\rho|\cos at|u.$$

Аналогічно до того, як це робилось у попередньому прикладі, звідси виводимо рівність, яку має задовольняти функція розтягування часу $I(t)$:

$$\frac{1}{a}|\sin at||\cos bI(t)|=\frac{1}{b}|\cos at||\sin bI(t)|. \qquad (8.39)$$

Функцію $I(t)$ будуватимемо поетапно, починаючи з півінтервалу часу $\left[0,\frac{\pi}{2a}\right)$, де $\cos at=|\cos at|>0$, $|tgat|=tgat$. За визначенням, функція розтягування часу $I(t)$, зокрема, має бути строго монотонною та такою, що

$I(0)=0$, $I(t)\geq t$ при $t\geq 0$. Тому шукатимемо функцію $I(t)$, для якої $\cos bI(t)>0$ при $t\in\left[0,\frac{\pi}{2a}\right)$.

Поділивши обидві частини рівності (8.39) на додатний добуток $|\cos at||\cos bI(t)|$, отримаємо

$$\frac{1}{a}tgat=\frac{1}{b}|tgbI(t)| \tag{8.40}$$

Функція $tgat$ має розриви при $t=(2k-1)\frac{\pi}{2a}, k=1,2,\ldots$. На інтервалі $\left(0,\frac{\pi}{2a}\right)$ функція $tgat$ набуває додатних значень, при $t=0$ обертається в нуль.

Нехай $tgbI(t)\geq 0$, $t\in\left[0,\frac{\pi}{2a}\right)$, тоді формула (8.40) перетворюється на рівність $\frac{1}{a}tgat=\frac{1}{b}tgbI(t)$, звідки маємо

$$I(t)=\frac{1}{b}arctg\left(\frac{b}{a}tgat\right). \tag{8.41}$$

Тут під символом *arctg* розуміється головне значення арктангенса. Бачимо, що функція $I(t)$ (8.41) є диференційованою на $\left[0,\frac{\pi}{2a}\right)$. Знайдемо похідну цієї функції.

$$\dot{I}(t)=\frac{a^2}{a^2\cos^2 at+b^2\sin^2 at}=\frac{a^2}{a^2-(a^2-b^2)\sin^2 at}. \tag{8.42}$$

Оскільки $\dot{I}(t)>0$, $t\in\left(0,\frac{\pi}{2a}\right)$, $I(0)=0$, то $I(t)$ не спадає на проміжку $\left[0,\frac{\pi}{2a}\right)$.

Таким чином, на півінтервалі $[0,\pi/2a)$ побудована функція розтягування часу $I(t)$, яка має вигляд (8.41), і на півінтервалі часу $[0,\pi/2a)$ вона має неперервну похідну. Покладемо $I\left(\frac{\pi}{2a}\right)=\frac{\pi}{2b}$.

Тепер проаналізуємо співвідношення (8.39) на півінтервалі $(\pi/2a,\pi/a]$, де $tgat<0$, і тому формула (8.39) перетворюється на рівність

$$\left|tgbI(t)\right|=-\frac{b}{a}tgat.$$

Слушно розглянути випадок, коли $tgbI(t)<0$, тоді $I(t)$ знаходиться з формули $tgbI(t)=\frac{b}{a}tgat$, звідки $I(t)$ при $t\in\left[\frac{\pi}{2a},\frac{\pi}{a}\right)$ теж має вигляд (8.41), при цьому на півінтервалі $\left(\frac{\pi}{2a},\frac{\pi}{a}\right)$ виконуються умова про монотонне неспадання функції $I(t)$ і умова $I(t)>t$. Отже, на півінтервалі часу $\left[0,\frac{\pi}{a}\right)$ побудовано функцію розтягування часу $I(t)$, що має вигляд (8.41) і має там неперервну похідну.

Аналогічно проаналізуємо співвідношення (8.39) на наступному півінтервалі неперервності $\left[\frac{\pi}{a},\frac{2\pi}{a}\right)$. Щоб виконувалась рівність $I\left(\frac{\pi}{a}\right)=\frac{\pi}{b}$, яка забезпечує неперервне продовження функції $I(t)$ на цей проміжок часу, скористаємось багатозначністю функції арктангенса і покладемо

$$I(t)=\frac{\pi}{b}+\frac{1}{b}arctg\left(\frac{b}{a}tgat\right),\ t\in\left[\frac{\pi}{a},\frac{2\pi}{a}\right).$$

З аналогічних міркувань отримаємо

$$I(t)=\frac{1}{b}\left[(k-1)\pi+arctg\left(\frac{b}{a}tgat\right)\right],\ t\in\left[(k-1)\pi/a,k\pi/a\right),\ \ k=1,2,\ldots \quad (8.43)$$

Дослідимо поведінку цієї функції. Бачимо, що функція $I(t)$ є неперервною функцією на півосі часу $[0,+\infty)$. Зауважимо, що похідна $\dot{I}(t)$ на відрізку часу $\left[0,\frac{\pi}{2a}\right]$ зростає від одиниці до $\frac{a^2}{b^2}$, а на $\left[\frac{\pi}{2a},\frac{\pi}{a}\right]$ спадає від $\frac{a^2}{b^2}$ до одиниці.

Це має місце для довільної пари відрізків

$$\left[(k-1)\frac{\pi}{a},(2k-1)\frac{\pi}{2a}\right],\ \left[(2k-1)\frac{\pi}{2a},k\frac{\pi}{a}\right],\ \ k=1,2,\ldots$$

Множина $W_1(t)$ (8.38) може бути представлена в такому вигляді:

$$W_1(t)=\rho\bigcup_{s\in R^n,\|s\|\le 1}\begin{pmatrix}\frac{1}{a}|\sin at|s\\ |\cos at|s\end{pmatrix}\stackrel{*}{-}\sigma\dot{I}(t)\bigcup_{s\in R^n,\|s\|\le 1}\begin{pmatrix}\frac{1}{a}|\sin bI(t)|s\\ |\cos bI(t)|s\end{pmatrix}. \qquad (8.44)$$

Згідно з визначенням (8.43) функція $I(t)$ задовольняє рівняння

$$tgbI(t)=\frac{b}{a}tgat.$$

Після його диференціювання отримаємо

$$\cos^2 bI(t)=\dot{I}(t)\cos^2 at.$$

З останніх двох рівностей випливають співвідношення

$$\begin{aligned}|\cos bI(t)|&=\sqrt{\dot{I}(t)}\,|\cos at|,\\ \frac{1}{b}|\sin bI(t)|&=\frac{1}{a}\sqrt{\dot{I}(t)}\,|\sin at|.\end{aligned} \qquad (8.45)$$

Друга рівність виконується, оскільки

$$\frac{1}{b}|\sin bI(t)|=\frac{1}{b}\sqrt{\sin^2 bI(t)}=\frac{1}{b}\sqrt{\left(tgbI(t)\cdot\cos bI(t)\right)^2}=$$

$$=\frac{1}{b}\sqrt{\frac{b^2}{a^2}tg^2at\cdot\dot{I}(t)\cos^2 at}=\frac{1}{a}\sqrt{\dot{I}(t)}\,|\sin at|.$$

Аналогічно до співвідношень (8.20), (8.21) для модельної гри, що досліджувалась раніш, формули (8.45) свідчать про виконання умови факторизації (8.22):

$$\pi e^{At}U=k(t)\pi e^{AI(t)}V.$$

Згідно з (8.45) тут $k(t) = \left(\dot{I}(t)\right)^{1/2}$.

Перепишемо формулу для $W_1(t)$ (8.44) з урахуванням рівностей (8.45):

$$W_1(t) = \left(\rho - \sigma\left(\dot{I}(t)\right)^{3/2}\right)\overline{S}(t),$$

де множина $\overline{S}(t)$ має вигляд:

$$\overline{S}(t) = \bigcup_{s\in R^n,\,\|s\|\le 1}\begin{pmatrix}\frac{1}{a}|\sin at|s\\ |\cos at|s\end{pmatrix}.$$

З формули для похідної $\dot{I}(t)$ (8.42) видно, що, оскільки $a > b$, то

$$\sup_{t\in[0,+\infty)} \dot{I}(t) = \frac{a^2}{b^2}.$$

Тому множина $W_1(t)$ містить у собі множину

$$\left(\rho - \sigma a^3/b^3\right)\overline{S}(t).$$

Звідси випливає, що наступні обмеження на параметри гри

$$a > b,\ \ \frac{\rho}{a^3} \ge \frac{\sigma}{b^3} \tag{8.46}$$

забезпечують виконання умови (8.38) про непорожність множини $W_1(t)$ (яка є умовою 8.2 для гри, що розглядається).

Покажемо, що за умови виконання нерівностей (8.46), переслідувач, будуючи своє керування з керування втікача в минулому, може здійснити м'яку зустріч із втікачем за довільних початкових положеннях та швидкостях супротивників. Для цього треба довести існування моменту часу t_1, визначеного формулою (8.8).

Опишемо спосіб керування переслідувача, що забезпечує існування такого моменту часу. На початковому відрізку часу покладемо керування переслідувача тотожно рівним нулю, тобто $u^0(\theta) \equiv 0$, $\theta \in [0, \tau_0)$. Тоді згідно з

(8.8) шуканий момент часу t_1 це той момент часу t, коли вперше виповниться включення

$$\begin{pmatrix} \cos\big(bI(t)\big)E & \frac{1}{b}\sin\big(bI(t)\big)E \\ -b\sin\big(bI(t)\big)E & \cos\big(bI(t)\big)E \end{pmatrix} \cdot \begin{pmatrix} y_0 \\ \dot{y}_0 \end{pmatrix} -$$

$$- \begin{pmatrix} \cos\big(aI(t)\big)E & \frac{1}{a}\sin\big(aI(t)\big)E \\ -a\sin\big(aI(t)\big)E & \cos\big(aI(t)\big)E \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ \dot{x}_0 \end{pmatrix} \in \int_0^t W_1(\theta)\,d\theta. \quad (8.47)$$

Матриці, що стоять у лівій частині цього включення та діють на вектори початкових станів об'єктів $(x_0, \dot{x}_0)$, $(y_0, \dot{y}_0)$, є операторами обертання. Тому вектор, що стоїть у лівій частині включення, з ростом t не залишатиме певну кульку rS, $rS \in R^{2n}$, радіуса r, $r > 0$, з центром у початку координат.

З попередніх міркувань випливає, що множина, яка стоїть у правій частині (8.47), містить у собі множину

$$\overline{W_1}(t) = \left(\rho - \sigma\frac{a^3}{b^3}\right) S(t),$$

де

$$S(t) = \bigcup_{s \in R^n,\, \|s\| \le 1} \begin{pmatrix} \frac{1}{a}\int_0^t \big|\sin a\theta\big|\,d\theta \cdot s \\ \int_0^t \big|\cos a\theta\big|\,d\theta \cdot s \end{pmatrix}.$$

Представимо поточний час t у вигляді: $t = k\frac{\pi}{2a} + \Delta$, $k = 0,1,2,\ldots$, де $0 \le \Delta < \frac{\pi}{2a}$. Тоді справедливі оцінки

$$\int_0^t \frac{1}{a}\big|\sin a\theta\big|\,d\theta = \frac{1}{a}\sum_{i=1}^{k} \int_{(i-1)\frac{\pi}{a}}^{i\frac{\pi}{a}} \big|\sin a\theta\big|\,d\theta + \frac{1}{a}\int_{k\frac{\pi}{a}}^{k\frac{\pi}{a}+\Delta} \big|\sin a\theta\big|\,d\theta \ge \frac{k\pi}{a^2},$$

$$\int_0^t \left|\cos a\theta\right| d\theta = \sum_{i=1}^{k} \int_{(i-1)\frac{\pi}{a}}^{i\frac{\pi}{a}} \left|\cos a\theta\right| d\theta + \int_{k\frac{\pi}{a}}^{k\frac{\pi}{a}+\Delta} \left|\cos a\theta\right| d\theta \geq \frac{k\pi}{a}.$$

Звідси доходимо висновку, що множина $\overline{W_1}(t)$ містить у собі кулю в просторі R^{2n} радіуса $k\pi \Big/ a\left(\rho - \frac{a^3}{b^3}\sigma\right)\frac{\sqrt{a^2+1}}{a}$ з центром у початку координат, яка при $k \to +\infty \left(t \to +\infty\right)$ прямує до кулі нескінченного радіуса з центром у нулі. Тому в деякий скінчений момент часу t_1 множина $\overline{W_1}(t)$ поглине кульку rS і виповниться включення (8.47).

Отже, якщо виконані умови (8.46) на параметри гри, то переслідувач, обираючи керування відповідно до формули (8.10), досягне м'якої зустрічі з втікачем у скінчений момент часу, який обчислюється на самому початку гри.

8.6. Висновки

Показано, що лінійна диференціальна гра зближення зі змінним запізненням інформації еквівалентна деякій грі з повною інформацією, що відкрило можливість застосування відомих методів до ігор із запізненням інформації.

На цій основі розвинутий принцип розтягування часу для вирішення ігор з повною інформацією, для яких не виконана умова Понтрягіна, зокрема в рамках першого прямого методу одержані достатні умови завершення таких ігор за скінчений час, при цьому вказаний спосіб керування переслідувача з огляду на керування втікача в пройдешній час, що знаходиться за допомогою функції розтягування часу.

Результати проілюстровані на складних задачах м'якої зустрічі об'єктів другого порядку, для яких функція розтягування часу знайдена в явному вигляді та виведені прості умови на параметри гри, за яких гра може бути завершена з будь-яких початкових станів.

Список літератури

1. Понтрягин Л. С. Избранные научные труды. М.: Наука, 1988. Т. 2. 576 с.
2. Chikrij G. Ts. An approach to the solution of linear differential games with variable information delay. Journal of Automation and Information Sciences. Beggel House, Inc. (USA). 1995. 27 (3&4). P. 163–170.
3. Chikrii A. A. Conflict-Controlled Processes. Berlin: Springer Science and Business Media, 2013. 424 p.
4. Никольский М. С. О применении первого прямого метода в линейных дифференциальных играх. Изв. АН СССР. Сер. техн. кибернетики. 1972. № 10. С. 51–56.
5. Зонневенд Д. Об одном методе преследования. ДАН СССР. 1972. Т. 204. № 6. С. 1296–1299.
6. Chikrii G. Ts. One approach to solution of complex game problems for some quasilinear evolutionary systems. Journal of Mathematics, Game Theory and Algebra. Nova Science Publishers, Inc. 2004. Vol. 14, N 4. P. 307–314.
7. Чикрий Г. Ц. Использование эффекта запаздывания информации в дифференциальных играх преследования. Кибернетика и системный анализ. 2007. № 2. С. 90–105.
8. G. Ts. Chikrii. Principle of time stretching in evolutionary games of approach. Journal of Automation and Information Sciences. 2016. N 5. P. 12–26.
9. G. Ts. Chikrii. Pontryagin's condition and its time stretching modification. Materials of the International Conference «Opimal Control and Differential Games» dedicated to the 110 th anniversary of L .S. Pontryagin (December 12−14, 2018, Steklov Mathematical Institute of RAS, Moscow).
10. G. Ts. Chikrii. Principle of time extension in dynamic games. Збірник матеріалів міжнародної наукової конференції «Сучасні проблеми математики та їх застосування в природничих науках і інформаційних технологіях». Чернівці, 2018. С. 126.

11. A. A. Chikrii, G. Ts. Chikrii, V. J. Zhukovskij, W. Wójcik and M. Junisbekov. Game problems of control for functional-differential systems. Recent Advances in Information Technology. Taylor & Francis Group, CRC Press, 2018. P. 13–50.
12. А. А. Чикрий, Г. Ц. Чикрий. Игровые задачи сближения для квазилинейных систем общего вида. Труды института математики и механики УрО РАН. 2018. Т. 24. № 1. С. 273–287.
13. Aumann R. J. Integrals of set-valued functions. J. Math. Anal. Appl. 1965. 12. P. 1–12.
14. Филиппов А. Ф. Дифференциальные уравнения с разрывной правой частью. Москва: Наука, 1985.
15. Василенко Н. В. Теория колебаний. Киев: Вища школа. 1992. 430 с.

ЗМІСТ

ВСТУП 3

РОЗДІЛ 1. СУБГРАДІЄНТНІ АЛГОРИТМИ З ПЕРЕТВОРЕННЯМ ПРОСТОРУ

1 Теорія та програмні реалізації r-алгоритмів Шора 5

П. І. Стецюк, Т. В. Бєлих, О. І. Криворучко

1.1 Вступ 5

1.2 Три обчислювальні форми r-алгоритмів 7

1.3 Три теореми про збіжність r-алгоритмів 14

1.4 $r(\alpha)$-Алгоритм з адаптивним кроком та його програмні реалізації .. 20

1.5 Octave-функція ralgb5a 25

1.6 Висновки 30

Список літератури 31

2 Субградієнтні методи з кроком Поляка та програма amsg2p 34

П. І. Стецюк, В. О. Стовба

2.1 Вступ 34

2.2 Позначення та визначення 35

2.3 Субградієнтний метод Поляка 37

2.4 Субградієнтний метод з кроком Поляка в перетвореному просторі . 45

2.5 Метод та програма amsg2p 53

2.6 Висновки 60

Список літератури 60

3 Узагальнений метод еліпсоїдів 62

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

3.1 Вступ 62

3.2 Загальна схема методу еліпсоїдів 64

3.3 H-форма узагальненого методу еліпсоїдів 69

3.4 Задачі для узагальненого методу еліпсоїдів 71

3.5 Алгоритм Шора та програма emshor ... 74
3.6 Висновки ... 80
Список літератури ... 81

РОЗДІЛ 2. ЗАДАЧІ НА КОМБІНАТОРНИХ КОНФІГУРАЦІЯХ

4 Задачі про математичні сейфи ... 83
Г. П. Донець
4.1 Вступ ... 83
4.2 Математичні сейфи на графах ... 84
4.3 Однотипні математичні сейфи з простим числом станів замків ... 89
4.4 Математичні сейфи з довільною кількістю станів замків ... 95
4.5 Математичні сейфи з різними типами замків ... 97
4.6 Висновки ... 110
Список літератури ... 110

5 Екстремальні задачі на комбінаторних конфігураціях ... 111
Г. П. Донець, Е. І. Ненахов
5.1 Побудова графа комбінаторної конфігурації ... 111
5.2 Оптимізація лінійної та квадратичної функцій на перестановках ... 119
5.3 Оптимізація дробово-лінійної функції на комбінаторних конфігураціях ... 128
5.4 Оптимізація лінійної та квадратичної функцій на комбінаторних конфігураціях з обмеженнями ... 137
Список літератури ... 148

6 Методи комбінаторного розпізнавання ... 149
Г. П. Донець, В. І. Білецький
6.1 Вступ ... 149
6.2 Обмежене комбінаторне розпізнавання ... 150
6.3 Необмежене комбінаторне розпізнавання ... 154
6.4 Розпізнавання двох радіоактивних куль ... 158
6.5 Деякі результати пошуку трьох та чотирьох активних куль ... 167

6.6 Висновки 168
Список літератури 169

РОЗДІЛ 3. КВАДРАТИЧНІ ТА ІГРОВІ ЗАДАЧІ

7 Про точні двоїсті оцінки для квадратичних екстремальних задач 170
О. А. Березовський
7.1 Вступ 170
7.2 Необхідна та достатня умова точної двоїстої оцінки для квадратичної екстремальної задачі 172
7.3 Необхідна та достатня умова точної двоїстої оцінки для квадратичної екстремальної задачі в матричному вигляді 178
7.4 Достатня умова точної двоїстої оцінки для квадратичної екстремальної задачі 182
7.5 Висновки 197
Список літератури 197

8 Принцип розтягування часу при прийнятті рішень в умовах конфлікту та невизначеності 199
Г. Ц. Чикрій
8.1 Вступ 199
8.2 Ефект запізнення інформації в лінійних диференціальних іграх 201
8.3 Використання ефекту запізнення інформації в лінійних диференціальних іграх зближення з повною інформацією 205
8.4 М'яке зближення керованих систем другого порядку, що описують ньютонівську динаміку за наявності тертя 210
8.5 М'яке зближення керованих систем другого порядку, що описують динаміку математичних маятників 221
8.6 Висновки 229
Список літератури 230

Наукове видання

П. І. Стецюк, Г. П. Донець, Е. І. Ненахов, Г. Ц. Чикрій,
О. А. Березовський, Т. В. Бєлих, В. І. Білецький, О. М. Хом'як,
О. О. Жмуд, А. В. Івлічев, О. І. Криворучко, В. О. Стовба

СУБГРАДІЄНТНІ АЛГОРИТМИ ТА ЗАДАЧІ НА КОМБІНАТОРНИХ КОНФІГУРАЦІЯХ

За загальною редакцією
доктора фізико-математичних наук П. І. Стецюка

В авторській редакції

Підписано до друку 26.06.2019. Зам. № 37-019
Гарнітура «Таймс». Друк офсетний. Папір офсетний.
Формат 60x84$^{1}/_{16}$. Обл.-вид. арк. 7,25. Ум. друк. арк. 6,4.
Наклад 100 пр.

Віддруковано: ТОВ «Університетське видавництво ПУЛЬСАРИ»
04070, Київ, вул. Спаська, 9/2
тел. (044) 425-12-75
e-mail: mail@pulsary.com.ua
pulsary.com.ua

Свідоцтво про внесення суб'єкта видавничої справи
до Державного реєстру видавців,
виготівників та розповсюджувачів видавничої продукції
серії ДК № 4436 від 08.11.2012

ISBN 978-617-615-093-0