УДК 51.681.3

С.Л. КРИВИЙ
Київський національний університет імені Тараса Шевченка, Київ, Україна,
e-mail: *sl.krivoi@gmail.com*.

О.В. ЧУГАЄНКО
ТОВ «SAMSUNG RND Ukraina», Київ, Україна, e-mail: *firestreami13@yahoo.com*.

АЛГОРИТМИ ПОБУДОВИ МІНІМАЛЬНОЇ ПОРОДЖУВАЛЬНОЇ МНОЖИНИ РОЗВ'ЯЗКІВ СИСТЕМ ЛІНІЙНИХ РІВНЯНЬ

Анотація. Розглянуто оптимізаційні перетворення алгоритму побудови мінімальної породжувальної множини розв'язків систем лінійних однорідних рівнянь у множині натуральних чисел. Описано особливості таких систем лінійних однорідних рівнянь, обґрунтовано оптимізаційні перетворення та наведено приклади роботи алгоритму до і після застосування оптимізаційних перетворень. Застосування алгоритму проілюстровано прикладами аналізу властивостей мереж Петрі та побудови множини базисних розв'язків у полях комплексних, дійсних, раціональних чисел та у скінченних полях.

Ключові слова: системи лінійних рівнянь, алгоритми, розв'язки, оптимізація, складність.

Є багато задач, які зводяться до розв'язання систем лінійних рівнянь (СЛР) [1, 2], що зумовило появу багатьох методів їхнього розв'язання. Особливе місце у цьому переліку посідають системи лінійних однорідних рівнянь (СЛОР), коефіцієнти яких належать кільцю цілих чисел Z, а їхні розв'язки шукають у множині натуральних чисел N. Більше століття тому було встановлено, що СЛОР такого типу мають скінченний базис множини всіх розв'язків [3, 4], але алгоритму побудови такого базису не було (тоді ще не існувало і самого поняття «алгоритм»). Дослідження з пошуку алгоритмів розв'язання СЛОР велися постійно, але особлива активність в цьому напрямі почалася у 80-х роках 20-го століття у зв'язку з роботами з побудови пруверів для теорій першого порядку. Побудова пруверів потребувала розв'язання проблеми уніфікації термів у асоціативно-комутативних теоріях першого порядку, яка зводилася до розв'язання СЛОР у множині N. Це зумовило появу алгоритмів розв'язання одного лінійного однорідного рівняння (ЛОР) [5], СЛОР [6–9] тощо. В кінці 20-го століття було розроблено TSS-метод і алгоритм розв'язання СЛОР у множині N для аналізу мереж Петрі (МП) [10]. Алгоритм, побудований на TSS-методі, описано в монографії [11]. Він застосовний для розв'язання систем лінійних обмежень типу рівностей і нерівностей. Особливістю цього алгоритму є те, що під час пошуку розв'язків СЛР не використовують операцію ділення за винятком скорочення на найбільший спільний дільник. Алгоритм знаходить мінімальну породжувальну множину розв'язків СЛОР у множині N, яка не завжди збігається з базисом множини всіх розв'язків СЛОР. Слід зазначити, що всі алгоритми розв'язання СЛР у множині N і TSS-алгоритм у тому числі мають високу часову складність [11, 12]. Але коли TSS-алгоритм застосовують до СЛР, розв'язки яких шукають у полі, то він має поліноміальну оцінку часової складності. Ця робота є продовженням [13].

1. TSS-АЛГОРИТМ ДЛЯ СЛОР У МНОЖИНІ N

Нехай задано СЛОР з цілими коефіцієнтами

$$S = \begin{cases} L_1(x) = a_{11}x_1 + \ldots + a_{1q}x_q = 0, \\ L_2(x) = a_{21}x_1 + \ldots + a_{2q}x_q = 0, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ L_p(x) = a_{p1}x_1 + \ldots + a_{pq}x_q = 0, \end{cases} \tag{1}$$

© С.Л. Кривий, О.В. Чугаєнко, 2024

де $a_{ij} \in Z$ (множина цілих чисел), а x_j набувають значень у множині N. Розглянемо множину одиничних векторів $M_0' = \{e_1, e_2, \ldots, e_q\}$ простору N^q, яку називають канонічним базисом цього простору, і перше рівняння $L_1(x) = a_{11}x_1 + a_{12}x_2 + \ldots + a_{1q}x_q = 0$ системи S. За допомогою функції $L_1(x)$ розіб'ємо елементи множини M_0' на такі три групи: $M_1^0 = \{e \mid L_1(e) = 0\}$, $M_1^+ = = \{e \mid L_1(e) > 0\}$ і $M_1^- = \{e \mid L_1(e) < 0\}$. Зрозуміло, що коли множини M_1^0 і M_1^+ чи M_1^0 і M_1^- порожні, то рівняння $L_1(x) = 0$ не має нетривіальних розв'язків у множині N. Якщо множини M_1^0, M_1^+, M_1^- непорожні, то побудуємо множину

$$M_1' = M_1^0 \cup \{y_{ij} \mid y_{ij} = -L_1(e_i)e_j + L_1(e_j)e_i\},$$

де $e_j \in M_1^+$, $e_i \in M_1^-$.

Використовуючи функцію $L_2(x)$, розіб'ємо елементи множини M' аналогічно до попереднього розбиття на три групи: $M_2^0 = \{e \mid L_2(e) = 0\}$, $M_2^+ = = \{e \mid L_2(e) > 0\}$ і $M_2^- = \{e \mid L_2(e) < 0\}$. Припустимо, що принаймні дві з цих множин непорожні, тоді побудуємо множину

$$M_2' = M_2^0 \cup \{y_{ij} \mid y_{ij} = -L_2(e_i)e_j + L_2(e_j)e_i, \text{ де } e_j \in M_2^+, e_i \in M_2^-\}.$$

Нехай у такий спосіб побудовано множину M_j' із множин $M_j^0 = = \{e_r^0 \mid L_j(e_r^0) = 0\}$, $M_j^+ = \{e_i^+ \mid L_j(e_i^+) > 0\}$ і $M_j^- = \{e_s^- \mid L_j(e_s^-) < 0$ за допомогою функції $L_j(x)$ і ця множина непорожня. Справедливою є очевидна теорема.

Теорема 1. Елементи множини M_j' є розв'язками системи рівнянь $L_1(x) = 0 \wedge L_2(x) = 0 \wedge \ldots \wedge L_j(x) = 0$.

Доведення очевидним чином випливає з побудови множини M_j'.

Означення 1. Побудована множина M_j' називається зрізаною множиною розв'язків або TSS (truncated set of solutions).

Ця назва себе виправдовує, оскільки отримана множина розв'язків не завжди буде базисом множини всіх розв'язків системи.

Породжувальна властивість TSS випливає з такої теореми.

Теорема 2. Нехай множина M — це TSS системи лінійних однорідних рівнянь (1) у множині N, побудована в описаний вище спосіб, і M' — множина всіх розв'язків цієї системи. Тоді довільний розв'язок $x \in M'$ можна представити у вигляді лінійної комбінації $tx = a_1e_1' + \ldots + a_me_m'$, де $t \geq 1$, $a_i \in N$, $e_i \in M$, $i = 1, \ldots, m$ [11].

Нехай (x, y) означає скалярний добуток векторів x і y, а $L_i(x)$ — ліву частину i-го рівняння системи S. Зі сказаного вище випливає такий алгоритм побудови TSS для СЛОДР у множині натуральних чисел N.

TSSEQN(M, p, q)

Вхід: Матриця СЛОДР S розміру $p \times q$.

Вихід: TSS-множина M розв'язків СЛОДР або «несумісна».

Метод:

$M := \{e_1, \ldots, e_q \mid e_i$ — канонічний вектор $N^q\}$;

for $i := 1$ to p do

$M := TSSEQN1(M, L_i(x))$;

if $M = \varnothing$ then (print(несумісна); STOP) else $CLEAN(M)$;

print(M);

TSSEQN1 ($M, L(x)$)

begin

$M^0 := \varnothing$; $M^+ := \varnothing$; $M^- := \varnothing$;

forall $e \in M$ do

if $L(e) = 0$ then $M^0 := M^0 \cup \{e\}$ else

if $L(e) > 0$ then $M^+ := M^+ \cup \{e\}$ else $M^- := M^- \cup \{e\}$;

$M' := M^0$;

if $M^+ \neq \varnothing \wedge M^- \neq \varnothing$ then

forall $u \in M^+$ do

forall $v \in M^-$ do $e := -L(v)u + L(u)v$; $M' := M' \cup \{e\}$;

return (M');

end

Процедура $CLEAN(M)$ виконує чищення отриманої множини розв'язків M. Це чищення зводиться до пошуку невід'ємних розв'язків СЛОР, які є невід'ємними лінійними комбінаціями решти розв'язків із множини M, і вилучення таких векторів із M.

З теорем 1, 2 випливає, що кожний вектор TSS можна поділити на найбільший спільний дільник (НСД) його координат, якщо цей НСД відмінний від одиниці. Це дає змогу зменшити величину координат цих векторів і ефективніше проводити обчислення.

Розглянемо приклади, які ілюструють роботу алгоритму **TSSEQN** та властивості множини розв'язків, отриманих цим алгоритмом. Покажемо спочатку, що результати обчислень TSS залежать від порядку рівнянь у системі.

Приклад 1. Перевірити сумісність СЛОДР S у множині N, де

$$S = \begin{cases} L_1(x) = 4x_1 + 2x_2 - 3x_3 - 2x_4 - x_5 = 0, \\ L_2(x) = 2x_1 - x_2 - 4x_3 + x_4 + 5x_5 = 0, \\ L_3(x) = 0x_1 + 2x_2 + 4x_3 + 0x_4 - 9x_5 = 0. \end{cases}$$

Розв'язання. TSS-множину M_1' складають вектори:

$$e_1 = (3, 0, 4, 0, 0),\ e_2 = (0, 3, 2, 0, 0),\ e_3 = (1, 0, 0, 2, 0),$$

$$e_4 = (0, 1, 0, 1, 0),\ e_5 = (1, 0, 0, 0, 4),\ e_6 = (0, 1, 0, 0, 2).$$

Базис множини розв'язків рівняння $L_1(x) = 0$ (який можна знайти, наприклад, методом з роботи [5]) складають вектори:

$$e_1 = (3, 0, 4, 0, 0),\ e_2 = (0, 3, 2, 0, 0),\ e_3 = (1, 0, 0, 2, 0),\ e_4 = (0, 1, 0, 1, 0),$$

$$e_5 = (1, 0, 0, 0, 4),\ e_6 = (0, 1, 0, 0, 2),\ e_7 = (2, 0, 2, 1, 0),\ e_8 = (1, 1, 2, 0, 0),$$

$$e_9 = (1, 0, 0, 1, 2),\ e_{10} = (0, 2, 1, 0, 1),\ e_{11} = (1, 0, 1, 0, 1).$$

На підставі теореми 2 вектори $e_7 - e_{11}$ цього базису повинні мати розклади через вектори $e_1 - e_6$. Дійсно,

$$2e_7 = e_1 + e_3,\ 3e_8 = e_1 + e_2,\ 2e_9 = e_3 + e_5,\ 2e_{10} = e_2 + e_6,\ 4e_{11} = e_1 + e_5.$$

Знаходячи значення функції $L_2(x)$ на векторах із M_1', маємо

$$L_2(e_1) = -10,\ L_2(e_2) = -11,\ L_2(e_3) = 4,\ L_2(e_4) = 0,\ L_2(e_5) = 22,\ L_2(e_6) = 9.$$

Будуємо

$$e'_1 = (0,1,0,1,0),\ e'_2 = 2e_1 + 5e_3 = (11,0,8,10,0),\ e'_3 = 4e_2 + 11e_3 = (11,12,8,22,0),$$

$$e'_4 = 2e_2 + e_5 = (1,6,4,0,4),\ e'_5 = 9e_2 + 11e_6 = (0,19,9,0,11),$$

$$e'_6 = 11e_1 + 5e_5 = (19,0,22,0,10),\ e'_7 = 9e_1 + 10e_6 = (27,10,36,0,20).$$

Якщо з отриманої множину розв'язків вилучити лінійно залежні розв'язки, то матимемо таку множину векторів:

$$M'_2 = \{e'_1 = (0,1,0,1,0),\ e'_2 = (11,0,8,10,0),\ e'_4 = (0,19,9,0,11),\ e'_5 = (19,0,22,0,10)\},$$

оскільки $e'_3 = e'_2 + 12e'_1$, $19e'_4 = 6e'_5 + e'_6$, $19e'_7 = 10e'_5 + 27e'_6$ і їх можна вилучити із TSS.

Базис множини розв'язків підсистеми $L_1(x) = 0 \wedge L_2(x) = 0$ складають такі вектори:

$$s_1 = (0,1,0,1,0),\ s_2 = (11,0,8,10,0),\ s_3 = (16,1,19,0,9),\ s_4 = (0,19,9,0,11),$$

$$s_5 = (19,0,22,0,10),\ s_6 = (1,6,4,0,4),\ s_7 = (13,2,16,0,8),\ s_8 = (10,3,13,0,7),$$

$$s_9 = (7,4,10,0,6),\ s_{10} = (4,5,7,0,5),\ s_{11} = (3,0,3,1,1).$$

Вектори $s_6 - s_{11}$ і s_3 є лінійними комбінаціями векторів

$$s_1 = (0,1,0,1,0),\ s_{2=(11,}\ 0,8,10,0),\ s_4 = (0,19,9,0,11),\ s_5 = (19,0,22,0,10),$$

і мають такий вигляд:

$$19s_6 = 6s_4 + s_5,\ 19s_7 = 2s_4 + 13s_5,\ 19s_8 = 3s_3 + 10s_5,$$

$$19s_9 = 4s_4 + 7s_5,\ 19s_{10} = 5s_4 + 4s_5,\ 10s_{11} = s_2 + s_5,\ 19s_3 = s_4 + 16s_5.$$

Значення $L_3(s_1) = 2$, $L_3(s_2) = 32$, $L_3(s_4) = -25$, $L_3(s_5) = -2$ свідчать про сумісність системи, яка має принаймні чотири розв'язки:

$$s'_1 = (19,1,22,1,10),\ s'_2 = (275,608,488,250,352),$$

$$s'_3 = (0,63,18,25,22),\ s'_4 = (63,0,72,2,32).$$

Але розв'язки s'_1, s'_2 лінійно виражаються через $s'_3 = (0,63,18,25,22)$, $s_4' = (63,0,72,2,32)$. Справді,

$$63(19,1,22,1,10) = 19(63,0,72,2,32) + (0,63,18,25,22),$$

$$63(275,608,488,250,352) = 275(63,0,72,2,32) + 608(0,63,18,25,22).$$

Отже, розв'язки

$$s'_3 = (0,63,18,25,22),\ s'_4 = (63,0,72,2,32)$$

складають TSS системи S.

Базис множини розв'язків цієї СЛОДР складають вектори

$$u_1 = s'_3 = (0,63,18,25,22),\ u_2 = s'_4 = (63,0,72,2,32),$$

$$u_3 = (19,1,22,1,10),\ u_4 = (13,4,16,2,8),\ u_5 = (7,7,10,3,6),\ u_6 = (1,10,4,4,4).$$

Отже, TSS-алгоритм не завжди будує базис множини всіх розв'язків СЛОДР.

Якщо поміняти перше рівняння з третім і застосувати TSS-алгоритм, то він знайде тільки розв'язки s'_3, s'_4.

З наведеного прикладу видно, що перед застосуванням TSS-алгоритму варто вибрати першим те рівняння СЛОР, у якого величина $kn + m$ мінімальна, де k — кількість додатних коефіцієнтів, n — кількість від'ємних коефіцієнтів і m — кількість нульових коефіцієнтів.

2. ОПТИМІЗАЦІЙНІ ПЕРЕТВОРЕННЯ TSS-АЛГОРИТМУ

Розглянемо оптимізаційні перетворення TSS-алгоритму та оцінимо їхню ефективність на конкретних прикладах СЛОР.

Перше оптимізаційне перетворення випливає з аналізу прикладу 1. Це перетворення зводиться до вибору першого рівняння системи, з якого починається побудова TSS. Вибираємо серед рівнянь системи те рівняння, у якого величина $kn + m$ мінімальна. Це перетворення дає певний ефект, як показують експерименти, але воно не завжди можливе в ситуації, коли для всіх рівнянь системи величина $kn + m$ однакова. Але це перетворення ставить запитання: чи можна величину $kn + m$ зменшити? Виявляється, що можна.

Друге оптимізаційне перетворення зводиться до діагоналізації матриці СЛОР. Діагоналізація має поліноміальну оцінку складності і дає змогу зменшити величину $kn + m$, про яку йшлося в першому перетворенні. Простий приклад ілюструє ефективність цього перетворення: нехай матриця A СЛОДР має такий вигляд:

$$A = \begin{pmatrix} 2 & -1 & 3 & 1 & -4 & 2 \\ 1 & 0 & -2 & -3 & 2 & 1 \\ -3 & 1 & 1 & 0 & -1 & 2 \\ 4 & 1 & -1 & -2 & 0 & 1 \end{pmatrix}, \quad A' = \begin{pmatrix} 0 & 0 & 0 & 31 & 5 & -38 \\ 0 & 0 & 2 & -2 & -3 & 5 \\ 0 & -2 & 0 & 25 & 5 & -15 \\ 1 & 0 & 0 & -5 & -1 & 6 \end{pmatrix}.$$

Перетворивши матрицю A до неповного трикутно-діагонального вигляду A' і застосувавши TSS-алгоритм, отримаємо таку кількість комбінацій $-L(v)u + L(u)v$, які обчислюються під час побудови TSS (TSS складається з двох розв'язків (16, 15, 89, 0, 76, 10) і (2, 3, 5, 7, 5)) цієї СЛОР з матрицями A і A': для першої матриці обчислюється 31 комбінація, а для другої — 8 комбінацій.

Але, як показують експериментальні результати, це оптимізаційне перетворення не завжди дає ефект. У разі розрідженої матриці системи, зокрема коли нульових коефіцієнтів у рівняннях системи більше половини кількості невідомих у цих рівняннях, то діагоналізацію матриці системи не варто застосовувати (див. експериментальні дані для систем 3 і 4 з табл. 1, матрицю системи 2 наведено у додатку).

Третє оптимізаційне перетворення випливає з додаткового аналізу векторів-розв'язків TSS з прикладу 1. Вектори, які входять до TSS, показують, що обчислення, пов'язані з наступним рівнянням системи, збільшують кількість ненульових координат у розв'язках TSS на одиницю. Якщо кількість ненульових координат у лінійній комбінації $e := -L(v)u + L(u)v$ у разі обчислень для i-го рівняння системи перевищує величину $q - (i+1)$ (q — кількість невідомих системи), то будувати цю комбінацію немає потреби. Обґрунтуванням цього факту є таке твердження.

Твердження 1. Якщо TSS підсистеми з перших j рівнянь містить вектор u, у якого кількість ненульових координат більша $q - (j+1)$, то в TSS знайдеться принаймні один вектор з меншою кількістю тих самих ненульових координат, що й у вектора u.

Доведення. Не обмежуючи загальності, розглянемо СЛОР з двома рівняннями. Нехай, наприклад, у результаті комбінування векторів-розв'язків

$$s_1 = (s_{11}, 0, \ldots, 0, s_{1i}, 0, \ldots, 0) \text{ і } s_2 = (0, s_{12}, 0, \ldots, 0, s_{1j}, 0, \ldots, 0),$$

з ненульовими координатами першого рівняння, де $i \neq j$, $s_{1i} < 0$, $s_{1j} < 0$, $s_{11} > 0$, $s_{12} > 0$, отримали вектор-розв'язок

$$u = -L_2(s_1)s_2 + L_2(s_2)s_1,\ L_2(s_1) = a < 0,\ L_2(s_2) = b > 0.$$

Але тоді крім векторів-розв'язків s_1 і s_2 першого рівняння в TSS цього рівняння повинні входити вектори $s_3 = (s_{11}, 0, \ldots, 0, s_{1j}, 0, \ldots, 0)$ і $s_4 = (0, s_{12}, 0, \ldots \ldots, 0, s_{1i}, 0, \ldots, 0)$. Можливі такі випадки:

1) $L_2(s_3) < 0$, тоді в TSS СЛОР входить вектор $v = -L_2(s_3)s_2 + L_2(s_2)s_3$, який має ненульові координати

$$v = (0, s_{12}, 0, \ldots, 0, s_{1i}, 0, \ldots, 0, s_{1j}, 0, \ldots, 0);$$

2) $L_2(s_3) > 0$, тоді в TSS СЛОР входить вектор $v = -L_2(s_1)s_3 + L_2(s_3)s_1$, який має ненульові координати

$$v = (s_{11}, 0, \ldots, 0, s_{1i}, 0, \ldots, 0, s_{1j}, 0, \ldots, 0).$$

Можливі випадки для другого вектора s_4 дають підстави для таких самих висновків. Отже, якщо вектор u потрапляє до TSS, то в TSS існує принаймні один вектор, який має ті самі ненульові координати, що й вектор u, але таких ненульових координат у нього менше, ніж у вектора u. Твердження доведено.

Обґрунтуванням того, що такий вектор можна вилучити з TSS, є така теорема.

Теорема 3. Нехай S — СЛОДР вигляду (1) і M'_p — її TSS, яка має k елементів. Тоді довільний вектор x із M'_p такий, що $tx > e_i \in M'_p \setminus \{x\}$, $t \in N$, і $t > 0$, можна представити у вигляді невід'ємної лінійної комбінації

$$mx = b_1 e_1 + b_2 e_2 + \ldots + b_{k-1} e_{k-1},$$

де $m \in N$, $m > 0$, $b_i \in N$, $i = 1, 2, \ldots, k-1$ [11].

Доведення. Якщо $tx \geq e_i \in M'_p \setminus \{x\}$, то розглянемо вектор $y = tx - a_i e_i$, де a_i — максимально можливе натуральне, відмінне від 0 число, для якого вектор y має невід'ємні координати. Якщо вектор $y \geq e_j \in M'_p \setminus \{x\}$, то розглянемо вектор $y_1 = y - a_i e_i - a_j e_j$, одержаний в аналогічний спосіб із векторів y і e_j. У результаті цих дій на деякому кроці отримаємо вектор y_r, щодо якого можуть виникнути такі два випадки:

а) $y_r = e_s \in M'_p \setminus \{x\}$;

б) y_r не належить $M'_p \setminus \{x\}$ і не порівнюється з жодним його елементом.

У випадку а) одержуємо шуканий розклад вектора tx:

$$y_r - e_s = 0 = tx - a_i e_i - a_j e_j - \ldots - a_r e_r - e_s$$

або $tx = a_i e_i + a_j e_j + \ldots + a_r e_r + e_s$.

У випадку б) внаслідок теореми 2 і того, що y_r є розв'язком системи S, маємо

$$k' y_r = b_1 e_1 + \ldots + b_{k-1} e_{k-1} + b_k x,$$

де $k' \neq 0$, $b_i \in \mathcal{N}$, і не всі b_i, $i = 1, 2, \ldots, k-1$, і b_k дорівнюють 0 одночасно. Але тоді вектор

$$k' y_r = k' tx - k' a_i e_i - k' a_j e_j - \ldots - k' a_r e_r = b_1 e_1 + \ldots + b_{k-1} e_{k-1} + b_k x$$

або

$$(tk' - b_k)x = b_1 e_1 + \ldots + (b_i + k'a_i)e_i + \ldots + (b_j + k'a_j)e_j + \ldots$$

$$\ldots + (b_r + ka_r)e_r + \ldots + b_{k-1}e_{k-1}$$

і є шуканим розкладом вектора x. Справді, оскільки хоча б один із коефіцієнтів a_i, b_j $(i, j = 1, \ldots, k-1)$ відмінний від 0, то $tk' \neq b_k$ і $tk' > b_k$. ■

У прикладі 1 під час знаходження TSS підсистеми з перших двох рівнянь не будуть обчислюватися розв'язки e'_3, e'_4, e'_7, а під час обчислення TSS всієї системи не будуть обчислюватися розв'язки s'_1, s'_2. Це прискорює обчислення TSS.

Обчислень комбінацій такого типу можна уникнути, якщо завчасно підрахувати кількість ненульових координат у комбінації $e := -L(v)u + L(u)v$. Результатом цього є така модифікація підпрограми **TSSEQN1** $(M, L(x))$:

TSSEQN1 $(M, L(x))$

begin

$M^0 := \varnothing$; $M^+ := \varnothing$; $M^- := \varnothing$;

forall $e \in M$ do

if $L(e) = 0$ then $M^0 := M^0 \cup \{e\}$ else

if $L(e) > 0$ then $M^+ := M^+ \cup \{e\}$ else $M^- := M^- \cup \{e\}$;

$M' := M^0$;

if $M^+ \neq \varnothing \wedge M^- \neq \varnothing$ then

forall $u \in M^+$ do (* де $u = (u(1), \ldots, u(q))$ *)

forall $v \in M^-$ do (* де $v = (v(1), \ldots, v(q))$ *)

$z := 0$; (* підрахунок кількості ненульових координат *)

for $j := 1$ to q do if $(u(j) \neq 0 \vee v(j) \neq 0)$ then $z := z + 1$ od

if $z \leq (q - (i+1))$ then $(e := -L(v)u + L(u)v$; $M' := M' \cup \{e\})$;

return (M');

end

2.1. Експериментальні результати оптимізаційних перетворень. Наведемо приклади конкретних СЛОР і результати застосування пропонованих оптимізаційних перетворень у табл. 1 (матриці СЛОР наведено в Додатку).

Загалом перше перетворення, якщо воно застосовне, дає в середньому 15–20 % прискорення обчислень, друге перетворення дає в середньому 25–35 % і третє перетворення — 20–25 %.

2.2. Застосування до аналізу мереж Петрі (МП). Одним із важливих засобів аналізу МП є множини інваріантів переходів і місць, які характеризують поведінку МП у процесі функціонування. Побудова множини інваріантів МП зводиться до пошуку розв'язків рівняння стану цієї МП у множині N. Рівняння стану є системою лінійних однорідних рівнянь. МП успішно застосовують для моделювання різних дискретних процесів, оскільки вони є виконáнними специфікаціями. Наведемо приклад МП, яка моделює симплексний протокол передачі даних у комп'ютерній мережі для зашумлених каналів з позитивним підтвердженням і повторним передаванням (протокол 3 з монографії [14]), і виконаємо аналіз властивостей цієї МП.

Сценарій протоколу має такий вигляд.

1. Мережевий рівень машини A передає пакет 1 своєму рівню передавання даних. Пакет доставляється цілим і збереженим на машину B і передається далі мережевому рівню. Машина B посилає кадр підтвердження назад на машину A.

Таблиця 1

Номер системи	Розмірності систем	Часові характеристики до застосування підрахунку нулів (с)	Часові характеристики після застосування підрахунку нулів (с)	Загальна кількість кандидатів	Кількість відкинутих кандидатів
1	12×25 до діагоналізації	92.27	79.73	36713861	5925852 (16.14 %)
2	12×25 після діагоналізації	11.23	8.52	4200874	1044861 (24.87 %)
3	20×41 до діагоналізації	60.41	50.00	11112408	3161878 (28.45 %)
4	20×41 після діагоналізації	1313.21	823.76	324660602	125703386 (38.72 %)
5	16×41 до діагоналізації	791.26	730.07	120595812	1178368 (0.98 %)
6	16×41 після діагоналізації	658.57	636.01	40400271	0 (0.00 %)
7	22×71 до діагоналізації	2218.84	2303.60	86415137	0 (0.00 %)
8	22×71 після діагоналізації	2569.78	2581.09	118942193	0 (0.00 %)

2. Кадр підтвердження повністю втрачається в каналі зв'язку і не потрапляє на машину А. Простіше було б, якби втрачалися тільки інформаційні кадри, а не кадри керування, але канал зв'язку їх не розрізняє.

3. У рівня передавання даних машини А раптово закінчується відведений інтервал часу (time-out). Не отримавши підтвердження, цей рівень вважає, що надісланий ним кадр з даними було зіпсовано або загублено, і надсилає цей кадр ще раз.

4. Дублікат кадру прибуває на рівень передачі даних машини В і передається на мережевий рівень. Якщо машина А надіслала на машину В файл, то частина цього файлу продублювалася і копія файлу на машині В буде помилковою. Інакше кажучи, протокол припустився помилки.

5. Отже, потрібен деякий механізм, за допомогою якого отримувач зможе відрізнити новий кадр від переданого повторно. Очевидним способом розв'язання цієї проблеми є розміщення відправником порядкового номера кадру в заголовку цього кадру. Тоді за номером кадру отримувач зможе зрозуміти, чи це новий кадр, чи його дублікат.

Для номера можна відвести лише один біт зі значенням 1 і 0.

Мережу Петрі, яка моделює цей сценарій протоколу, наведено на рис. 1.

Переходи t_1, t_2 відповідають звичайному і повторному передаванню кадру 0. Переходи t_3, t_4 означають те саме для кадру 1. Переходи t_5, t_6, t_7 відповідають втраті кадру 0, втраті підтвердження і втраті кадру 1. Переходи t_8, t_9 означають надходження кадру з неправильним номером. Переходи t_{10}, t_{11} означають отримання машиною, що приймає, наступного кадру та передачу його мережевому рівню.

Протокол повинен виконувати умову: жодна послідовність переходів не може мати двох входжень переходу t_{10}, не маючи між цими входженнями переходу t_{11}. Верифікація такої умови виконується шляхом обчислення інваріантів переходів цієї МП. Для цього розглянемо рівняння стану наведеної МП, яка має сім рівнянь і 11 невідомих (за кількістю станів і переходів у МП):

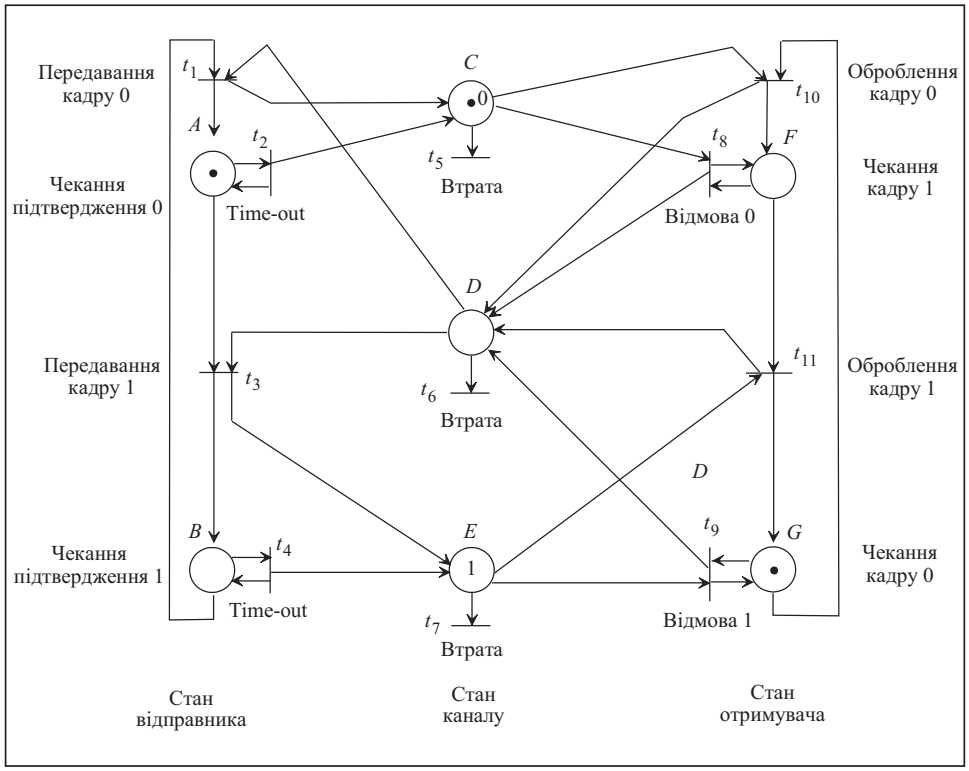

Рис. 1

$$Ax=\begin{cases} A & 1x_1 & 0x_2 & -1x_3 & 0x_4 & 0x_5 & 0x_6 & 0x_7 & 0x_8 & 0x_9 & 0x_{10} & 0x_{11} \\ B & -1x_1 & 0x_2 & 1x_3 & 0x_4 & 0x_5 & 0x_6 & 0x_7 & 0x_8 & 0x_9 & 0x_{10} & 0x_{11} \\ C & 1x_1 & 1x_2 & 0x_3 & 0x_4 & -1x_5 & 0x_6 & 0x_7 & -1x_8 & 0x_9 & -1x_{10} & 0x_{11} \\ D & -1x_1 & 0x_2 & -1x_3 & 0x_4 & 0x_5 & -1x_6 & 0x_7 & 1x_8 & 1x_9 & 1x_{10} & 1x_{11} \\ E & 0x_1 & 0x_2 & 1x_3 & 1x_4 & 0x_5 & 0x_6 & -1x_7 & 0x_8 & -1x_9 & 0x_{10} & -1x_{11} \\ F & 0x_1 & 0x_2 & 0x_3 & 0x_4 & 0x_5 & 0x_6 & 0x_7 & 0x_8 & 0x_9 & 1x_{10} & -1x_{11} \\ G & 0x_1 & 0x_2 & 0x_3 & 0x_4 & 0x_5 & 0x_6 & 0x_7 & 0x_8 & 0x_9 & -1x_{10} & 1x_{11} \end{cases} = 0$$

Розв'язки цього рівняння називають інваріантами переходів МП. Ці інваріанти свідчать про те, що МП жива (всі переходи спрацьовують) і що протокол працює правильно. Дійсно, спрацювання переходу t_{10} завжди веде до спрацювання переходу t_{11} (інваріанти третій і дев'ятий). Інваріанти переходів МП наведено у табл. 2.

Таблиця 2

t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}
0	1	0	0	1	0	0	0	0	0	0
1	0	1	0	0	0	0	1	1	0	0
1	0	1	0	0	0	0	0	0	1	1
0	1	0	0	0	1	0	1	0	0	0
0	0	0	1	0	0	1	0	0	0	0
0	0	0	1	0	1	0	0	1	0	0
1	0	1	1	1	0	0	0	2	0	0
1	1	1	0	0	0	1	2	0	0	0
0	1	0	1	0	2	0	0	0	1	1

3. TSS-АЛГОРИТМ ДЛЯ СЛР В ЧИСЛОВИХ ПОЛЯХ

Розглянемо застосування TSS-методу для розв'язання СЛР у полі дійсних чисел $\mathcal{D}$. Складність алгоритмів оцінюють за кількістю операцій, які потрібно виконати в найгіршому випадку.

Нехай задано лінійне однорідне рівняння (ЛОР)

$$a_1x_1 + a_2x_2 + \ldots + a_qx_q = 0, \tag{2}$$

де $a_i \in \mathcal{D}$, $i = 1, 2, \ldots, q$.

Не обмежуючи загальності припустимо, що $a_1 \neq 0$, тоді, комбінуючи цей коефіцієнт з рештою ненульових коефіцієнтів, побудуємо таку множину векторів:

$$s_1 = (a_2, -a_1, 0, 0, \ldots, 0, 0),\ s_2 = (a_3, 0, -a_1, 0, \ldots, 0, 0), \ldots,$$

$$s_{q-2} = (a_{q-1}, 0, 0, 0, \ldots, -a_1, 0),\ s_{q-1} = (a_q, 0, 0, 0, \ldots, 0, -a_1).$$

Очевидно, що побудовані вектори є розв'язками рівняння (2). Якщо деякі коефіцієнти в ЛОР дорівнюють нулю, то ця множина розв'язків поповнюється відповідними векторами канонічного базису.

Лема 1. Множина *TSS* є базисом множини розв'язків ЛОР (2). Складність побудови множини *TSS* є $O(q^2)$, де q — кількість невідомих у рівнянні.

Доведення. Нехай $s = (b_1, b_2, \ldots, b_q)$ — довільний розв'язок (2), тоді

$$s + \frac{b_2}{a_1}s_1 + \frac{b_3}{a_1}s_2 + \ldots + \frac{b_q}{a_1}s_{q-1} = \left(b_1 + \frac{a_2b_2 + a_3b_3 + \ldots + a_qb_q}{a_1}, 0, 0, \ldots, 0\right) =$$

$$= \left(\frac{a_1b_1 + a_2b_2 + a_3b_3 + \ldots + a_qb_q}{a_1}, 0, 0, \ldots, 0\right) = (0, 0, \ldots, 0)$$

на підставі того, що s — розв'язок рівняння (2). Звідси отримуємо

$$s = -\frac{b_2}{a_1}s_1 - \frac{b_3}{a_1}s_2 - \ldots - \frac{b_q}{a_1}s_{q-1},$$

а це на підставі довільності s означає, що множина *TSS* — базис множини розв'язків (2).

Складність побудови однієї комбінації потребує q кроків, а всіх таких векторів $q-1$. Отже, складність побудови всієї множини *TSS* буде $O(q^2)$. ■

Приклад 2. Знайти базиси множини розв'язків рівняння: $1.2x_1 + 1.5x_2 + 0.7x_3 + 0.4x_4 = 0$.

Згідно з доведеним вище, базисними розв'язками рівняння будуть вектори:

$$s_1 = (1.5, -1.2, 0, 0),\ s_2 = (0.7, 0, -1.2, 0),\ s_3 = (0.4, 0, 0, -1.2).$$

Наприклад, подання розв'язку $s = (4, -1, 1, -10)$ рівняння в базисі його розв'язків набуває вигляду $s = \frac{1}{1.2}s_1 - \frac{1}{1.2}s_2 + \frac{10}{1.2}s_3$.

Нехай маємо СЛОР S (1). Розглянемо множину векторів канонічного базису $M_0' = \{e_1, e_2, \ldots, e_q\}$ і перше рівняння $L_1(x) = a_{11}x_1 + a_{12}x_2 + \ldots + a_{1q}x_q = 0$ системи S. Побудуємо базис $B_1 = \{e_1', e_2', \ldots, e_m'\}$ множини всіх розв'язків цього ЛОР описаним вище способом. Візьмемо функцію $L_2(x) = a_{21}x_1 + \ldots + a_{2q}x_q$ і розглянемо ЛОР вигляду

$$L_2(e_1')y_1 + L_2(e_2')y_2 + \ldots + L_2(e_m')y_m = 0. \tag{3}$$

Зауважимо, що коли всі $L_2(e'_i)=0$, то рівняння $L_2(x)$ лінійно виражається через $L_1(x)$ і його можна видалити зі СЛОР S.

Знайдемо базис $B'=\{r_1, r_2, \ldots, r_k\}$ множини розв'язків ЛОР (3) в описаний вище спосіб і побудуємо за векторами з B' відповідні лінійні комбінації векторів із B_1. Позначимо цю множину $M=\{s_1, s_2, \ldots, s_k\}$.

Лема 2. Множина M — базис множини розв'язків

$$S=\begin{cases} L_1(x)=a_{11}x_1+\ldots+a_{1q}x_q=0, \\ L_2(x)=a_{21}x_1+\ldots+a_{2q}x_q=0. \end{cases} \tag{4}$$

Доведення. Очевидно, що всі елементи із M є розв'язками СЛОР S. Нехай $x=(x_1, \ldots, x_q)$ — довільний розв'язок СЛОР (4), тоді відповідно до леми 1

$$x=d_1e'_1+\ldots+d_me'_m,$$

де $e'_i \in B_1$, $i=1, \ldots, m$. Підставивши x в $L_2(x)$, отримаємо ЛОР

$$d_1L_2(e'_1)+\ldots+d_mL_2(e'_m)=c_1d_1+\ldots+c_md_m=0, \tag{5}$$

тобто вектор $(d_1, d_2, \ldots, d_m)$ є розв'язком ЛОР (5) і, отже, він є лінійною комбінацією векторів із B':

$$(d_1, \ldots, d_m)=f_1r_1+\ldots+f_kr_k$$

Але тоді

$$x=d_1e'_1+\ldots+d_me'_m=f'_1s_1+\ldots+f'_ks_k,$$

а це означає, що розв'язок x можна подати у вигляді лінійної комбінації векторів із M. На підставі довільності вектора x отримуємо справедливість леми. ■

Теорема 4. Нехай M — множина *TSS*, побудована в описаний вище спосіб для СЛОР S, тоді M є базисом множини всіх розв'язків цієї СЛОР. Складність побудови базису M становить $O(pq^2)$, де p — кількість рівнянь, а q — кількість невідомих в СЛОР.

Доведення виконуємо індукцією за кількістю k рівнянь у СЛОР S.

База індукції ($k=1$) має місце на підставі леми 1.

Припустимо, що теорема справедлива для всіх $k<p$. Тоді множина *TSS* розв'язків СЛОР S', яка складається з перших $p-1$ рівнянь, за припущенням індукції є базисом множини розв'язків S'.

Повторюючи хід викладок, які аналогічні тим, що використовувалися у доведенні леми 2, отримуємо справедливість теореми.

Оцінка часової складності, наведена у формулюванні теореми, очевидним чином випливає з побудови множини *TSS* і леми 1. ■

Приклад 3. Побудувати базис множини розв'язків СЛОР

$$S=\begin{cases} 2.1x_1+1.1x_2+0x_3+x_4+2.3x_5=0, \\ 0.5x_1+0.2x_2+1x_3+0x_4-1.3x_5=0, \\ 0.1x_1+0x_2+2.5x_3+2x_4+0x_5=0. \end{cases}$$

Розв'язання. Базис множини розв'язків першого ЛОР цієї системи:

$$s_1=(1.1, -2.1, 0, 0, 0),\ s_2=(0, 0, 1, 0, 0),$$

$$s_3=(1, 0, 0, -2.1, 0),\ s_4=(2.3, 0, 0, 0, -2.1).$$

Знаходимо $L_2(s_1)=0.13$, $L_2(s_2)=1$, $L_2(s_3)=0.5$, $L_2(s_4)=3.88$ і будуємом ЛОР $0.13y_1+1y_2+0.5y_3+3.88y_4=0$. Базис множини розв'язків цього ЛОР скла-

дають вектори $r_1 = (1, -0.3, 0, 0)$, $r_2 = (0.5, 0, -0.13, 0)$, $r_3 = (3.88, 0, 0, -0.13)$. Будуємо множину розв'язків перших двох рівнянь із S, що відповідають векторам r_1, r_2, r_3:

$$s'_1 = s_1 - 0.13s_2 = (1.1, -2.1, -0.13, 0, 0),$$

$$s'_2 = 0.5s_1 - 0.13s_3 = (0.42, -1.05, 0, 0.273, 0),$$

$$s'_3 = 3.88s_1 - 0.13s_4 = (3.969, -8{,}148, 0, 0, 0.273).$$

Знаходимо значення $L_3(s'_1) = -0.215$, $L_3(s'_2) = 0.588$, $L_3(s'_3) = 0.3969$, будуємо ЛОР $-0.215y_1 + 0.588y_2 + 0.3969y_3 = 0$ і знаходимо його розв'язки: $r'_1 = (0.588, 0.215, 0)$, $r'_2 = (0.3969\ 0, 0.215)$. Будуємо відповідні вектори базису множини розв'язків СЛОР S:

$$m_1 = 0.588s'_1 + 0.215s'_2 = (0.7371, -1.46055, -0.07644, 0.058695, 0),$$

$$m_2 = 0.3969s'_1 + 0.215s'_3 = (1.289925, -2.58531, -0.051597, 0, 0.058695).$$

Отримані розв'язки складають фундаментальну систему розв'язків цієї СЛОР. Для того, щоб її отримати в класичному вигляді, потрібно поділити координати знайдених розв'язків на 0.058695. Дійсно, в результаті такого ділення матимемо

$$f_1 = (12.55813953, -24.88372093, -1.302325581, 1, 0),$$

$$f_2 = (21.97674419, -44.04651163, -0.879069767, 0, 1).$$

При цьому виконано всього шість операцій ділення.

Оскільки під час побудови базису множини розв'язків не використовується операція ділення, то коли коефіцієнти системи є цілими числами, базисні розв'язки теж будуть мати цілі значення координат. Оскільки базисні розв'язки складають фундаментальну систему розв'язків, то звідси випливає лінійна незалежність векторів множини *TSS* та важлива властивість *TSS*-методу, а саме, його інкрементальність. Нехай $TSS(S_1 \wedge S_2)$ означає множину *TSS* розв'язків системи S, яка складається з двох підсистем S_1 і S_2, а $TSS(M_1, S_2)$ — множина *TSS* розв'язків підсистеми S_2 з початковою множиною *TSS* M_1 першої підсистеми, а $TSS(M_2, S_1)$ — множина *TSS* з початковою множиною *TSS* M_2 другої підсистеми. Тоді властивість інкрементальності виражається такими залежностями:

$$TSS(\varnothing, S_1 \wedge S_2) = TSS(M_1, S_2) = TSS(M_2, S_1),$$

де рівності справджуються з точністю до знака. Це очевидно випливає з властивостей детермінантів (у разі перестановки рядків системи детермінант змінює знак на протилежний).

Зі сказаного вище випливає такий інкрементальний алгоритм побудови базису множини розв'язків СЛОР у полі дійсних чисел $\mathcal{D}$.

TSSEQ (A, M, k, p, q)

Вхід: СЛОР з матрицею A розміру $p \times q$ і початковою множиною розв'язків M, k — номер рівняння СЛОР.

Вихід: Базис M множини розв'язків СЛОР або «несумісна».

Метод:

if $M = \varnothing$ then

(k:=1;

M:= $\{e_1, \ldots, e_q | e_i$ — канонічний вектор $N^q\}$);

for i:= k to p do

$M := TSSE1(M, L_{i(x)})$;
if $M = \varnothing$ then (print(несумісна);STOP) else print(M);

TSSE1$(M, L(x))$

begin

$M' := \varnothing$; $M^c := \varnothing$;
for $m = 1$ to $|M|$ do
pick e_m from M;
if $L(e_m) = 0$ then $M' := M' \cup \{e_m\}$ else $M^c := M^c \cup \{e_m\}$;
pick first e_j from M^c;
forall e_i from M^c do
$e_i := L(e_i)e_j - L(e_j)e_i$; $M' := M' \cup \{e_i\}$;
return(M');

end

Складність побудови базису, як випливає з теореми 4, пропорційна величині pq^2, тобто має оцінку $O(n^3)$, де $n = \max(p, q)$.

Розглянутий *TSS*-метод розв'язання СЛОР з незначними змінами застосовують до розв'язання СЛНР.

Нехай задано СЛНР

$$S = \begin{cases} L_1(x) = a_{11}x_1 + \ldots + a_{1q}x_q = b_1, \\ L_2(x) = a_{21}x_1 + \ldots + a_{2q}x_q = b_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ L_p(x) = a_{p1}x_1 + \ldots + a_{pq}x_q = b_p. \end{cases} \tag{6}$$

Редукуємо СЛНР 6) до вигляду

$$S' = \begin{cases} L_1(x) = a_{11}x_1 + \ldots + a_{1q}x_q - b_1x_0 = 0, \\ L_2(x) = a_{21}x_1 + \ldots + a_{2q}x_q - b_2x_0 = 0, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ L_p(x) = a_{p1}x_1 + \ldots + a_{pq}x_q - b_px_0 = 0, \end{cases} \tag{7}$$

де x_0 — додаткова невідома величина.

Нехай $B = \{s_1, s_2, \ldots, s_m\}$ — базис множини розв'язків СЛОР (7). Розіб'ємо множину B на дві підмножини

$$B_0 = \{s_i \in B : s_i = (d_{i1}, d_{i2}, \ldots, d_{iq}, 0\},$$

$$B_1 = \{s_j \in B : s_j = (d_{j1}, d_{j2}, \ldots, d_{jq}, d\},$$

де $d \neq 0$. Якщо $B_1 = \varnothing$, то СЛНР (6) несумісна, а коли $B_1 \neq \varnothing$ і в B_1 немає вектора, у якого $d = 1$, то поділимо всі координати якого-небудь вектора $s_j = (d_{j1}, \ldots, d_{jq}, d) \in B_1$ на число d. Отримаємо вектор $s'_j = \left(\frac{d_{j1}}{d}, \frac{d_{j2}}{d}, \ldots, \frac{d_{jq}}{d}, 1\right)$.

Тоді загальний розв'язок СЛНР (6) набуває вигляду

$$x = s''_j + c_1 s'_{i1} + \ldots + c_r s'_{ir},$$

де $s''_j = \left(\frac{d_{j1}}{d}, \frac{d_{j2}}{d}, \ldots, \frac{d_{jq}}{d}\right)$ — частинний розв'язок СЛНР (6), а $s'_{ik} = (d_{i1}, d_{i2}, \ldots, d_{iq})$ знаходять з векторів множини B_0 шляхом відкидання останньої нульової координати.

Дійсно, множині B_0 належать вектори, які складають базис множини всіх розв'язків супутньої СЛОР для СЛНР (6). А вектори із множини B_1 — це частинні

розв'язки СЛНР. За довільним вектором із множини B_1 за потреби будуємо вектор s''_j, в чому і полягають незначні зміни наведеного *TSS*-алгоритму.

Зауважимо, що побудова базису розв'язків СЛР у полі комплексних чисел зводиться до СЛР у полі дійсних чисел, а для поля раціональних чисел наведений алгоритм застосовують без змін.

У скінченних полях зміни алгоритму стосуються лише обчислень, оскільки все залишається без змін, тільки обчислення виконуються за модулем поля. У цьому випадку можна отримати точніші оцінки часової складності (так звану бітову складність), якщо зображувати коефіцієнти і проміжні значення двійковими словами.

ДОДАТОК

Наведемо для наочності приклади двох матриць, з використанням яких було проведено тестування оптимізаційних перетворень.

1. Матриця системи розміру 12×25:

$$\begin{pmatrix}
0 & -1 & -3 & 1 & 0 & 4 & -3 & -2 & 1 & -2 & 5 & 0 & 5 & -1 & 3 & -4 & -4 & 1 & -4 & 0 & -1 & -3 & -1 & 0 & -5 \\
-2 & -3 & 4 & -3 & 3 & 1 & 1 & 4 & -3 & -4 & -1 & 0 & -2 & 0 & 2 & 2 & -4 & 1 & -4 & -1 & 1 & 2 & 2 & 3 & 5 \\
2 & 3 & 2 & 0 & 3 & 2 & 5 & -4 & -5 & -3 & -3 & 3 & 3 & 5 & 0 & 2 & 1 & 5 & -3 & 0 & 2 & 5 & 2 & 4 & 1 \\
4 & -1 & 4 & -3 & -3 & -3 & 2 & 1 & 3 & 4 & -4 & 4 & 2 & 2 & -3 & -4 & -2 & 3 & -2 & 2 & -5 & 4 & 3 & 2 & 4 \\
-1 & -1 & -3 & 4 & -3 & 2 & -3 & -3 & 1 & 5 & 4 & -3 & -4 & 1 & 0 & -1 & 4 & -1 & -2 & 2 & 2 & -4 & 3 & -5 & 5 \\
3 & -3 & -1 & 3 & 4 & -3 & 1 & -3 & -2 & -1 & -3 & 4 & 3 & -3 & 4 & 3 & -5 & 0 & 2 & -1 & 4 & -3 & 1 & 2 & 4 \\
-4 & 2 & 4 & -2 & 5 & 3 & -2 & 1 & 5 & -4 & -2 & 1 & 4 & 1 & 5 & -3 & 3 & 4 & 3 & -2 & 5 & 0 & 4 & -2 & -2 \\
-1 & 3 & -1 & -4 & 0 & -4 & -4 & 0 & 3 & 1 & -3 & -1 & 3 & 1 & 5 & -1 & 4 & 0 & 4 & 5 & -4 & -3 & -1 & -3 & -2 \\
-1 & -1 & 1 & -2 & 0 & -3 & -4 & -5 & 3 & -1 & 4 & -3 & -1 & -4 & 1 & 4 & 4 & 2 & 4 & -3 & 5 & 0 & 4 & 3 & 2 \\
2 & -1 & 1 & -1 & -2 & 4 & -2 & -3 & 0 & 0 & -1 & 1 & 0 & -3 & -2 & 1 & 5 & 2 & -3 & -1 & -1 & -5 & -4 & 2 & -1 \\
3 & 3 & 4 & -4 & 0 & 4 & -3 & -1 & -3 & -4 & 3 & -3 & 1 & -5 & -3 & 2 & -4 & 4 & 0 & 3 & 1 & 0 & 2 & -2 & -2 \\
2 & -4 & 3 & 4 & 5 & -3 & 2 & 1 & 2 & -2 & -4 & -5 & 1 & 2 & 0 & 3 & 2 & 4 & 0 & 1 & -2 & 4 & 3 & -2 & 2
\end{pmatrix}.$$

2. Матриця системи розміру 20×41:

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

СПИСОК ЛІТЕРАТУРИ

1. Boumova S., Drensky V., Kostadinov V. A Diophantine transport problem from 2016 and its possible solution from 1903. arXiv:2003.06766v1 [math.NT] 15 Mar 2020. P. 1–25.
2. Stanley R. Linear homogeneous Diophantine equations and magic labellings of graphs. *Duke Math. Journ.* 1973. Vol. 40, Iss. 3. P. 607–632. https://doi.org/10.1215/S0012-7094-73-04054-4.

3. Gordan P. Ueber die Auflösung linearer Gleichungen mit reallen Coefficienten. *Mathematische Annalen*. 1873. Vol. 6. P. 23–28.

4 Hilbert D. Ueber die Theorie der algebraischen Formen. *Mathematische Annalen*. 1890. Vol. 36. P. 473–534.

5. Clausen M., Fortenbacher A. Efficient solution of linear Diophantine equations. *Journal of Symbolic Computation*. 1989. Vol. 8, Iss. 1–2. P. 201–216. https://doi.org/10.1016/S0747-7171(89)80025-2.

6. Contejean E., Devie H. An efficient incremental algorithm for solving systems of linear Diophantine equations. *Information and Computation*. 1994. Vol. 113, Iss. 1. P. 143–172. https://doi.org/10.1006/inco.1994.1067.

7. Domenjoud E. Outils pour la deduction automatique dans les theories associatives-commutatives: Thése de Doctorat d'Universite. Universite de Nancy I, 1991.

8. Pottier L. Minimal solution of linear Diophantine systems: bounds and algorithms. *Proc. 4th International Conference on Rewriting Techniques and Applications* (10–12 April 1991, Como, Italy). Como, 1991. P. 162–173.

9. Motroi V., Ciobaca St. A note on the performance of algorithms for solving linear Diophantine equations in the naturals. arXiv:2104.05200v1 [cs.DS] 12 Apr 2021. P. 1–11.

10. Крывый С.Л. О вычислении минимального набора инвариантов сети Петри. Искусственный интеллект. 2001. № 3. P. 199–206.

11. Кривий С.Л. Лінійні діофантові обмеження та їх застосування. Київ: Інтерсервіс, 2021. 260 с.

12. Hermann M., Juban L., Kolaitis P. G. On the complexity of counting the Hilbert basis of a linear Diophantine system. *LNCS*. 1999. Vol. 1705. P. 13–32.

13. Кривий С.Л., Чугаєнко О.В. Модифікація алгоритму побудови мінімальної генеруючої множини розв'язків СЛР у множині натуральних чисел. *Фізико-математичне моделювання та інформаційні технології.* 2023. № 36. С. 131–136.

14. Таненбаум Э. Компьютерные сети. 4-е изд. СПб.: Питер, 2003. 992 с.

S. Kryvyi, O. Chugaenko

ALGORITHMS FOR CONSTRUCTION OF MINIMAL GENERATING SET OF SOLUTIONS FOR SYSTEMS OF LINEAR EQUATIONS

Abstract. We consider optimizing transformations of the algorithm for construction of minimal generating sets of solutions of systems of linear homogeneous equations (SLHE) over the set of natural numbers. The features of such SLHEs are described, optimization transforms are substantiated, and examples of algorithm operation before and after optimization transforms are given. The application of the algorithm is illustrated by examples of the analysis of the properties of Petri nets and the construction of a set of basic solutions in the fields of complex, real, and rational numbers and over finite fields.

Keywords: systems of linear equations, algorithms, solutions, optimization, complexity.

Надійшла до редакції 03.11.2023