УДК 51.681.3

С.Л. Крывый, В.Т. Антонюк

АЛГОРИТМЫ РЕШЕНИЯ СИСТЕМ ЛИНЕЙНЫХ ОГРАНИЧЕНИЙ С ЦЕЛЫМИ КОЭФФИЦИЕНТАМИ ВО МНОЖЕСТВЕ {0, 1}

Ключевые слова: системы диофантовых уравнений, неравенств, базис множества решений, независимые множества вершин, ловушки и дедлоки сетей Петри.

Введение

Метод, предлагаемый в данной работе, называется TSS-методом, как и алгоритмы, которые реализуют этот метод. TSS-алгоритм решения систем линейных однородных уравнений (СЛОУ) вида

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

где $a_{ij} \in Z, x_i \in N, i = 1,\ldots, p, j = 1,\ldots, q,$ носит комбинаторный характер, в котором комбинируются в решение коэффициенты с противоположными знаками первого уравнения. Полученные таким образом решения подставляются в следующее уравнение, и по найденным значениям на этих решениях находятся их линейные комбинации тем же способом комбинирования. Полученные линейные комбинации являются решениями первого и второго уравнений. Дальше все это повторяется для третьего уравнения, четвертого и т.д. Полученное таким образом множество решений называется TSS.

Пример 1. Найти решения СЛОУ S во множестве натуральных чисел N

$$S = \begin{cases} 2x - y + z - u = 0, \\ -x + y + 2z - u = 0, \\ x + y + z - 2z = 0. \end{cases}$$

TSS первого уравнения включает такие решения:

$$s_1 = (1, 2, 0, 0),\ s_2 = (0, 1, 1, 0),\ s_3 = (1, 0, 0, 2),\ s_4 = (0, 0, 1, 1).$$

Значения левой части второго уравнения на этих решениях: 1, 3, –3, 1. По этим значениям находим линейные комбинации (комбинируя значение –3 с остальными значениями):

$$u_1 = 3s_1 + s_3 = (4, 6, 0, 2) \to (2, 3, 0, 1),$$
$$u_2 = s_2 + s_3 = (1, 1, 1, 2),\ u_3 = 3s_4 + s_3 = (1, 0, 3, 5).$$

Значения левой части третьего уравнения на этих решениях 3, –1, –6. По этим значениям находим линейные комбинации

$$v_1 = u_1 + 3u_2 = (5, 6, 3, 7) \text{ и } v_2 = 2u_1 + u_3 = (5, 6, 3, 7).$$

Это и есть TSS СЛОУ S. Но это решение вычисляется дважды. Если в системе переставить на первое место третье уравнение, то для третьего уравнения получаем такие решения:

$$s_1 = (2, 0, 0, 1),\ s_2 = (0, 2, 0, 1),\ s_3 = (0, 0, 2, 1).$$

На этих решениях значения второго уравнения: –3, 3, 1, комбинации по которым дают решения

$$u_1 = (1, 3, 0, 2),\ u_2 = (1, 0, 1, 1).$$

На этих решениях значения первого уравнения: –3, 2, комбинации по которым дают решение $2u_1 + 3u_2 = (5, 6, 3, 7)$. ♠

Отметим, что TSS-алгоритм описан в [1], поэтому здесь не приводится, а рассматриваются только его свойства и модификации.

Очевидно, что элементы TSS — решение СЛОУ, а порождающее свойство TSS вытекает из такого утверждения.

Теорема 1. Пусть $M'_j = \{e'_1, \ldots, e'_l\}$ — TSS системы S, а M_j — множество всех ее решений. Тогда любой вектор $x \in M_j \setminus M'_j$ представляется в виде неотрицательной линейной комбинации

$$tx = b_1 e'_1 + \ldots + b_l e'_l, \tag{2}$$

где $t, b_i \in N, t > 1, e'_i \in M'_j, i = 1, \ldots, l$ [1].

Из приведенного выше примера вытекает, что множество решений СЛОУ зависит от порядка уравнений в системе. Для того чтобы решения не зависели от этого порядка, в процессе вычислений нужно выполнять процедуру чистки. Имеет место следующая теорема.

Теорема 2. Пусть S — СЛОУ и M'_p — ее TSS, которое имеет k элементов. Тогда любой вектор x из M'_p такой, что $tx \geq e_i \in M'_p \setminus \{x\}, t \in N$ и $t \neq 0$, представляется неотрицательной линейной комбинацией

$$mx = b_1 e_1 + \ldots + b_{i-1} e_{i-1} + b_{i+1} e_{i+1} + \ldots + b_k e_k, \tag{3}$$

где $m \neq 0, m, b_i \in N, i = 1, 2, \ldots, k-1$ [1].

Из этой теоремы следует такая простая процедура чистки TSS: вектор x удаляется из TSS, если x или его произведение tx больше некоторого вектора из TSS. Множителем t можно взять, в частности, максимальную координату векторов из текущего TSS. Например, при решении системы S решение $u_2 = (1, 1, 1, 2)$ можно удалить, поскольку $5u_2$ больше u_1 и u_3. А представление этого решения в виде линейной комбинации (3) принимает вид

$$3(1, 1, 1, 2) = u_1 + u_3 = (2, 3, 0, 1) + (1, 0, 3, 5).$$

ISSN 0572-2691

Пусть СЛОУ S состоит из двух подсистем: S_1 и S_2, т.е. $S = S_1 \wedge S_2$ и $TSS(M, S)$ означают TSS для данной СЛОУ S с начальным TSS M. Если M совпадает с каноническим базисом, то будем писать $\mathrm{TSS}(\varnothing, S)$. Имеют место утверждения, доказательство которых приведено в [1].

Теорема 3. Пусть $M_1 = \mathrm{TSS}(\varnothing, S_1)$, $M_2 = \mathrm{TSS}(\varnothing, S_2)$, тогда $\mathrm{TSS}(\varnothing, S_1 \wedge S_2) = \mathrm{TSS}(M_1, S_2) = \mathrm{TSS}(M_2, S_1)$.

Следствие 1. Пусть S_1, S_2 — две СЛОУ и $\mathrm{TSS}(\varnothing, S_1) = M_1$, $\mathrm{TSS}(\varnothing, S_2) = M_2$:

а) $\mathrm{TSS}(\varnothing, S_1 \wedge S_2) = \mathrm{TSS}(\varnothing, S_2 \wedge S_1)$;

б) Если $\mathrm{TSS}(M_2, S_1) = M_2$, то $Sol(S_1) \subseteq Sol(S_2)$, где $Sol(S)$ — множество всех решений СЛОУ S;

в) S_1 и S_2 эквивалентны тогда и только тогда, когда $M_1 = M_2$.

Отсюда, в частности, следует, что в процессе построения TSS можно определять (и в случае необходимости удалять) линейно зависимые уравнения системы, а также вычислять ранг матрицы данной системы.

1. Численный алгоритм решения СЛОУ и неравенств в области {0, 1}

Существует общий метод решения систем линейных однородных неравенств (СЛОН) путем введения дополнительных неизвестных и сведения к решению СЛОУ. Анализ результатов экспериментов показал, что введение дополнительных неизвестных существенно влияет на эффективность алгоритма. Оказывается, что можно решать систему неравенств без введения дополнительных неизвестных.

Рассмотрим модификацию TSS-алгоритма построения базиса множества всех решений СЛОУ во множестве $\{0, 1\}$, который достаточно экономный по памяти и базируется на свойствах TSS, приведенных в теореме 3 и следствии 1.

1.1. Алгоритм построения базиса. Рассмотрим сначала СЛОУ вида

$$S = \begin{cases} 1x_1 + 0x_2 + \ldots + 0x_q + 1x_{q+1} + 0x_{q+2} + \ldots + 0x_{2q} - 1x_{2q+1} = 0, \\ 0x_1 + 1x_2 + \ldots + 0x_q + 0x_{q+1} + 1x_{q+2} + \ldots + 0x_{2q} - 1x_{2q+1} = 0, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ 0x_1 + 0x_2 + \ldots + 1x_q + 0x_{q+1} + 0x_{q+2} + \ldots + 1x_{2q} - 1x_{2q+1} = 0, \end{cases} \quad (4)$$

которая соответствует системе неравенств

$$S' = \begin{cases} x_1 \le 1, \\ x_2 \le 1, \\ \ldots \\ x_q \le 1. \end{cases}$$

Например, рассмотрим систему

$$S = \begin{cases} 1x_1 + 0x_2 + 0x_3 + 0x_4 + 1x_5 + 0x_6 + 0x_7 + 0x_8 - 1x_9 = 0, \\ 0x_1 + 1x_2 + 0x_3 + 0x_4 + 0x_5 + 1x_6 + 0x_7 + 0x_8 - 1x_9 = 0, \\ 0x_1 + 0x_2 + 1x_3 + 0x_4 + 0x_5 + 0x_6 + 1x_7 + 0x_8 - 1x_9 = 0, \\ 0x_1 + 0x_2 + 0x_3 + 1x_4 + 0x_5 + 0x_6 + 0x_7 + 1x_8 - 1x_9 = 0. \end{cases}$$

TSS данной системы совпадает с базисом множества всех решений и состоит из таких 16 векторов, первые четыре координаты которых отделены пропусками:

$x_1 = (1,1,1,1,\ 0,0,0,0,\ 1)$, $x_2 = (0,1,1,1,\ 1,0,0,0,\ 1)$, $x_3 = (1,0,1,1,\ 0,1,0,0,\ 1)$,

$x_4 = (0,0,1,1,\ 1,1,0,0,\ 1)$, $x_5 = (1,1,0,1,\ 0,0,1,0,\ 1)$, $x_6 = (0,1,0,1,\ 1,0,1,0,\ 1)$,

$x_7 = (1,0,0,1,\ 0,1,1,0,\ 1)$, $x_8 = (0,0,0,1,\ 1,1,1,0,\ 1)$, $x_9 = (1,1,1,0,\ 0,0,0,1,\ 1)$,

$x_{10} = (0,1,1,0,\ 1,0,0,1,\ 1)$, $x_{11} = (1,0,1,0,\ 0,1,0,1,\ 1)$, $x_{12} = (0,0,1,0,\ 1,1,0,1,\ 1)$,

$x_{13} = (1,1,0,0,\ 0,0,1,1,\ 1)$, $x_{14} = (0,1,0,0,\ 1,0,1,1,\ 1)$, $x_{15} = (1,0,0,0,\ 0,1,1,1,\ 1)$,

$$x_{16} = (0,0,0,0,\ 1,1,1,1,\ 1).$$

Следовательно, TSS-алгоритм генерирует для таких систем базис множества всех решений и состоит из 2^q векторов. Эти векторы имеют такие свойства:

а) первые q координат и вторые q координат дополняют одна другую, т.е. если i-я координата первой части равна 1 (0), то $(q+i)$-я координата второй части равна 0 (1), а последняя координата такого решения всегда равна 1;

б) величины первых q координат возрастают от 0 до $2^q - 1$, т.е. включают все возможные комбинации 0 и 1.

Приведенные свойства множества решений для такого типа СЛОУ позволяют построить экономный по памяти (но переборный по времени) алгоритм вычисления базиса множества всех решений для СЛОУ во множестве $\{0,1\}$. Этот алгоритм реализует полный перебор вариантов, не требуя введения дополнительных переменных. Действительно, рассмотрим систему уравнений и неравенств:

$$S' = \begin{cases} S_0, \\ x_1 \ \le\ 1, \\ x_2 \ \le\ 1, \\ \dots\ \dots\ \dots \\ x_q \ \le\ 1, \end{cases}$$

где S_0 — некоторая заданная СЛОУ.

Вводя дополнительные переменные и преобразовывая систему S' к системе уравнений, получаем такую СЛОУ:

$$S'' = \begin{cases} S_0', \\ 1\ \ 0\ \dots\ 0\ \ 1\ \ 0\ \dots\ 0\ -1\ =\ 0, \\ \dots\ \dots\ \dots\ \dots\ \dots\ \dots\ \dots\ \dots\ \dots\ \dots\ \dots\ \dots \\ 0\ \ 0\ \dots\ 1\ \ 0\ \ 0\ \dots\ 1\ -1\ =\ 0, \end{cases}$$

где S_0' — система S_0, дополненная нулевыми коэффициентами, которые соответствуют дополнительным переменным.

Из свойств TSS-метода следует, что TSS данной системы совпадает с TSS системы:

$$S_1 = \begin{cases} 1\ \ 0\ \dots\ 0\ \ 1\ \dots\ 0\ -1\ =\ 0, \\ \dots\ \dots\ \dots\ \dots\ \dots\ \dots\ \dots\ \dots\ \dots\ \dots \\ 0\ \ 0\ \dots\ 1\ \ 0\ \dots\ 1\ -1\ =\ 0, \\ S_0'. \end{cases}$$

Следовательно, в вычислении множества решений для подсистемы, которая состоит из уравнений предшествующих подсистеме S_0' (назовем ее верхней подсисте-

ISSN 0572-2691

мой), нет необходимости, потому что это множество известно. Поскольку уравнения подсистемы S_0 дополнены нулями, только первые q координат TSS верхней подсистемы влияют на окончательное TSS всей системы S_0 в целом. Так как закон построения этих координат известен, то при вычислении TSS нет необходимости вводить дополнительные переменные и запоминать все TSS векторы верхней подсистемы. TSS всей системы можно вычислять, перебирая последовательно все варианты векторов TSS верхней подсистемы, запоминая только базисные решения. Причем при таком переборе можно использовать разные порядки перебора.

Используя эти факты и покоординатный порядок на булевых векторах, приведем структуры данных алгоритма и сам алгоритм в терминах этих структур.

Пусть x — q-мерный булев вектор ($x(j)$ — его j-я координата), в котором запоминается текущее значение вектора-решения СЛОУ S_0, B обозначает базис множества всех решений S_0 во множестве $\{0,1\}$, $b(j)$ — j-й вектор-столбец матрицы A системы S_0. При таких структурах данных алгоритм принимает вид:

TSS-01($b(1), b(2), \ldots, b(q), p$)
Вход. $b_1, b_2, \ldots, b_q$ — столбцы матрицы A системы $A \times x = 0$, а p — их размерность.
Выход. B — базис множества решений системы $A \times x = 0$ в множестве $\{0,1\}$.
Метод.

```
begin
  B = Ø;  x = 0;
  for i = 1 to 2^(q-1) do
    x = x + 1; y = 0;
    for j = 1 to q do
      if x(j) = 1 then y = y + b(j)
    od
    if y = 0 then add x to B
    od
end
```

Добавление найденного решения (операция add x to B) выполняется таким образом: если x больше некоторого $y \in B$, то x не добавляется к B, иначе y удаляется из B и в B заносится решение x.

Характеристику временной сложности данного алгоритма дает следующая теорема.

Теорема 4. Временная сложность алгоритма TSS-01 $O(pq \cdot 2^{q-1})$, а память, которая требуется алгоритму, $O(\max(pq, |B|q))$.

Доказательство непосредственно следует из приведенного алгоритма TSS-01.

Отметим, что алгоритм применим к произвольной числовой области, поскольку при этом он абсолютно не зависит от того, какой числовой области принадлежат коэффициенты системы. Кроме того, алгоритм применим без каких-либо модификаций к неоднородным системам уравнений и системам неравенств.

Несмотря на высокую оценку временной сложности алгоритма TSS-01, он допускает распараллеливание. Действительно, множество векторов, которые проверяются, можно разбить на $k = 2^q / 2^r$ интервалов, где k — количество процессоров. Найдя решения на каждом из этих интервалов, комбинируем их в базис множества всех решений СЛОУ.

1.2. Приложение к специальным системам линейных неравенств. Рассмотрим системы линейных неравенств, коэффициенты которых принадлежат

множеству $\{-1, 0, 1\}$. Незначительная модификация TSS-алгоритма позволяет строить базис множества решений таких СЛОН в области $\{0, 1\}$. Для этого в процессе применения TSS-алгоритма необходимо находить решения не только уравнений, но и неравенств.

Пусть дано линейное однородное неравенство (ЛОН)

$$L(x) = a_1x_1 + a_2x_2 + \ldots + a_qx_q \geq 0, \tag{5}$$

где $a_i \in \{-1, 0, 1\}$, $x_i \in \{0,1\}$, $i = 1, 2, \ldots, q$. Для такого типа ЛОН TSS-алгоритм строит базис множества всех решений в области $\{0, 1\}$. Это построение выполняется таким образом. Комбинируем в решение векторы канонического базиса так, как это делалось выше. К этим решениям добавляем векторы канонического базиса, которые соответствуют нулевым и положительным коэффициентам ЛОН (5). Пусть $M' = M_0 \cup M_> \cup M^0 = \{s_1, s_2, \ldots, s_k\}$, где $M_0 = \{e_i \mid L(e_i) = 0\}$, $M_> = \{e_j \mid L(e_j) > 0\}$, и $M^0 = \{e_i + e_j \mid L(e_i) = 1, L(e_j) = -1\}$, таким образом построено множество решений ЛОН (5). Имеет место следующая теорема.

Теорема 5. Множество M' является базисом множества всех решений ЛОН (5) в области $\{0, 1\}$.

Доказательство. Пусть $a = (a_1, a_2, \ldots, a_q)$ — любое решение ЛОН (5) и $a_{i_1}, \ldots, a_{i_r}$ — все ненулевые координаты вектора a. Вычитая соответствующие векторы, принадлежащие множеству $M_0 \cup M^0$, удалим из множества ненулевых координат вектора a все те, которые соответствуют нулевым коэффициентам и которые соответствуют нулевым решениям ЛОН (5). Это значит, что построен вектор

$$a' = a - e_{j_1} - e_{j_2} - \ldots - e_{j_r}, \tag{6}$$

где $e_{j_l} \in M_0 \cup M^0$. Очевидно, что вектор a' также будет решением ЛОН (5). Если $a' = 0$, то теорема доказана. Если же $a' \neq 0$, то его ненулевые координаты должны соответствовать векторам из множества $M_>$. Поскольку TSS включает все векторы, которые соответствуют положительным коэффициентам, удалим эти координаты из вектора a' путем вычитания векторов из $M_>$. В результате получим вектор $a'' = a' - e_{l1} - \ldots - e_{lp}$. Покажем, что $a'' = 0$. Если $a'' \neq 0$ и $L(a'') > 0$, то это значит, что вектор a'' имеет по крайней мере одну ненулевую координату. Но это противоречит построению вектора a'' и построению множества $M_>$. Если $a'' \neq 0$ и $L(a'') = 0$, то это противоречит построению вектора a'. Следовательно, $a'' = 0$. Но тогда

$$a = a' + e_{j_1} + e_{j_2} + \ldots + e_{j_r} = e_{l_1} + \ldots + e_{l_k} + e_{j_1} + e_{j_2} + \ldots + e_{j_p},$$

что и нужно было доказать. ■

Из этой теоремы получаем очередное следствие.

Следствие 2. Множество M' является базисом множества всех решений линейно однородных равенств (ЛОР) $L(x) = a_1x_1 + a_2x_2 + \ldots + a_qx_q = 0$ в области $\{0, 1\}$ [1].

Действительно, в этом случае базисными решениями ЛОР будут векторы из $M_0 \cup M^0$, т.е. векторы канонического базиса, которые соответствуют нулевым коэффициентам, и векторы, которые получены в результате комбинирования коэффициентов с разными знаками ЛОР.

Пример 2. Для ЛОН

$$x_1 - x_2 + x_3 + x_4 + 0x_5 - x_6 \geq 0$$

базисными решениями соответствующего ЛОР будут векторы:

$$M^0 = \{s_1 = (1, 1, 0, 0, 0, 0),\ s_2 = (0, 1, 1, 0, 0, 0),\ s_3 = (0, 1, 0, 1, 0, 0),$$

$$s_4 = (1, 0, 0, 0, 0, 1),\ s_5 = (0, 0, 1, 0, 0, 1),\ s_6 = (0, 0, 0, 1, 0, 1)\},$$

$$M_0 = \{s_7 = (0, 0, 0, 0, 1, 0)\}.$$

К этим решениям добавляются векторы канонического базиса, которые соответствуют положительным коэффициентам ЛОН:

$$M_> = \{s_8 = (1, 0, 0, 0, 0, 0),\ s_9 = (0, 0, 1, 0, 0, 0),\ s_{10} = (0, 0, 0, 1, 0, 0)\}.$$

Все векторы составляют базис множества решений данного ЛОН. Очевидным решением ЛОН есть вектор $a = (1, 1, 1, 1, 0, 1)$, который имеет разложение

$$a = s_1 + s_5 + s_{10} = (1, 1, 1, 1, 0, 1).\ \spadesuit$$

Рассмотрим системы линейных однородных неравенств (СЛОН). Из теоремы 5 следует, что когда ищутся решения СЛОН, у которой значения следующего уравнения или неравенства на найденных решениях предыдущих уравнений или неравенств принадлежат множеству $\{-1, 0, 1\}$, то TSS-алгоритм находит базис множества всех решений СЛОН. Сначала рассмотрим СЛОН вида

$$S = \begin{cases} L_1(x) = a_{11}x_1 + \ldots + a_{1q}x_q \geq 0, \\ L_2(x) = a_{21}x_1 + \ldots + a_{2q}x_q \geq 0, \end{cases} \tag{7}$$

где $a_{ij}, \in \{-1, 0, 1\}$, $x_i \in \{0, 1\}$, $i = 1, 2$, $j = 1, ..., q$.

Построим базис $M = \{s_1, \ldots, s_m\}$ множества решений ЛОН $L_1(x) \geq 0$ описанным выше способом. Подставим эти базисные решения в другое неравенство и построим по найденным значениям ЛОН

$$L_2(s_1)\, y_1 + L_2(s_2)\, y_2 + \ldots + L_2(s_m)\, y_m \geq 0. \tag{8}$$

Поскольку все значения $L_2(s_i) \in \{-1, 0, 1\}$, пусть $M' = \{v_1, v_2, \ldots, v_k\}$ — базисные решения ЛОН (8). Построим линейные комбинации базисных векторов из M, которые соответствуют векторам из M', и удалим те комбинации, координаты которых больше 1. Кроме того, некоторые из полученных таким образом решений могут быть выражены как линейные комбинации остальных решений. Это иллюстрирует нижеприведенный пример.

Пример 3. Построить базис множества решений СЛОН:

$$S = \begin{cases} x_1 - x_2 + x_3 + x_4 + 0x_5 - x_6 \geq 0, \\ 0x_1 + x_2 - x_3 + 0x_4 - x_5 + x_6 \geq 0. \end{cases}$$

Решение. Базис множества решений первого неравенства найден в примере 2:

$$M^0 = \{s_1 = (1, 1, 0, 0, 0, 0),\ s_2 = (0, 1, 1, 0, 0, 0),\ s_3 = (0, 1, 0, 1, 0, 0),$$

$$s_4 = (1, 0, 0, 0, 0, 1),\ s_5 = (0, 0, 1, 0, 0, 1),\ s_6 = (0, 0, 0, 1, 0, 1)\},$$

$$M_0 = \{s_7 = (0, 0, 0, 0, 1, 0)\},$$

$$M_> = \{s_8 = (1, 0, 0, 0, 0, 0),\ s_9 = (0, 0, 1, 0, 0, 0),\ s_{10} = (0, 0, 0, 1, 0, 0)\}.$$

Значения на этих базисных решениях при подстановке в другое неравенство таковы: $L_2(s_1)=1, L_2(s_2)=0, L_2(s_3)=1, L_2(s_4)=1, L_2(s_5)=0, L_2(s_6)=1, \quad L_2(s_7)=-1,$ $L_2(s_8)=0, L_2(s_9)=-1, L_2(s_{10})=0.$ Строим неравенство

$$y_1+0y_2+y_3+y_4+0y_5+y_6-y_7+0y_8-y_9+0y_{10}\geq 0.$$

Решениями этого неравенства есть векторы:

$$y_1=(0,1,0,0,0,0,0,0,0,0),\ y_2=(0,0,0,0,1,0,0,0,0,0),$$

$$y_3=(0,0,0,0,0,0,0,1,0,0),\ y_4=(0,0,0,0,0,0,0,0,0,1),$$

$$y_5=(1,0,0,0,0,0,1,0,0,0),\ y_6=(0,0,1,0,0,0,1,0,0,0),$$

$$y_7=(0,0,0,1,0,0,1,0,0,0),\ y_8=(0,0,0,0,0,1,1,0,0,0),$$

$$y_9=(1,0,0,0,0,0,0,0,1,0),\ y_{10}=(0,0,1,0,0,0,0,0,1,0),$$

$$y_{11}=(0,0,0,1,0,0,0,0,1,0),\ y_{12}=(0,0,0,0,0,1,0,0,1,0),$$

$$y_{13}=(1,0,0,0,0,0,0,0,0,0),\ y_{14}=(0,0,1,0,0,0,0,0,0,0),$$

$$y_{15}=(0,0,0,1,0,0,0,0,0,0),\ y_{16}=(0,0,0,0,0,1,0,0,0,0).$$

Им соответствуют такие решения системы:

$$M_0=\{s_2, s_5, s_8, s_{10}\},\ \ M_{>}=\{s_1, s_3, s_4, s_6\},$$

$$M^0=\{s_{17}=(1,1,0,0,1,0),\ s_{37}=(0,1,0,1,1,0),\ s_{47}=(1,0,0,0,1,1),$$

$$s_{67}=(0,0,0,1,1,1),\ s_{19}=(1,1,1,0,0,0),\ s_{39}=(0,1,1,1,0,0),$$

$$s_{49}=(1,0,1,0,0,1),\ s_{69}=(0,0,1,1,0,1).$$

Векторы $s_{19}, s_{39}, s_{49}, s_{69}$ представляются такими линейными комбинациями:

$$s_{19}=(1,1,1,0,0,0)=s_2+s_8=(0,1,1,0,0,0)+(1,0,0,0,0,0),$$

$$s_{39}=(0,1,1,1,0,0)=s_2+s_{10}=(0,1,1,0,0,0)+(0,0,0,1,0,0),$$

$$s_{49}=(1,0,1,0,0,1)=s_5+s_8=(0,0,1,0,0,1)+(1,0,0,0,0,0),$$

$$s_{69}=(0,0,1,1,0,1)=s_5+s_{10}=(0,0,1,0,0,1)+(0,0,0,1,0,0).\ \spadesuit$$

Возникает вопрос: как находить те решения, которые не входят в базисные решения? Метод поиска таких решений вытекает из предыдущего подраздела 1.1 и реализуется в два этапа:

• сравниваем выбранное решение a с остальными построенными решениями, и если $a>b$ некоторого решения b, то вычисляем векторы значений $(L_1(a), L_2(a))$ и $(L_1(b), L_2(b))$;

• если $(L_1(a), L_2(a))>(L_1(b), L_2(b)),$ то вектор a удаляем из текущего множества решений, иначе векторы a и b остаются.

Например, вектор $s_{19}>s_1$, но векторы $(L_1(s_{19}), L_2(s_{19}))=(1,0)$ и $(L_1(s_1),$ $L_2(s_1))=(0,1)$ не сравнимы между собой, а векторы $(L_1(s_{19}), L_2(s_{19}))=(1,0)$ и $(L_1(s_2), L_2(s_2))=(0,0)$ сравнимы, и поскольку первый больше второго, вектор s_{19} удаляется из множества решений системы.

 ISSN 0572-2691

Пусть $S = \{u_1, u_2, \ldots, u_r\}$ — построенное таким образом множество решений СЛОН.

Теорема 6. Множество S является базисом множества всех решений СЛОН (7) во множестве $\{0, 1\}$.

Доказательство. Пусть $a = (a_1, a_2, \ldots, a_l)$ — произвольное решение (7). Это решение, будучи решением ЛОН $L_1(x) \geq 0$, представляется в виде линейной комбинации базисных векторов из M:

$$a = c_1 s_{i1} + c_2 s_{i2} + \ldots + c_t s_{it}$$

и $L_2(a) = c_1 L_2(s_{i1}) + c_2 L_2(s_{i2}) + \ldots + c_t L_2(s_{it}) \geq 0$ в силу того, что a — решение СЛОН (7). Но тогда вектор $c = (c_1, c_2, \ldots, c_t)$, будучи решением (8), представляется в виде линейной комбинации базисных векторов из M'. Иными словами,

$$c = d_1 v_{i1} + d_2 v_{i2} + \ldots + d_t v_{it}$$

и $a = v_{i1} + v_{i2} + \ldots + v_{it}$, где $v_{ij} \in S$. Действительно, поскольку все координаты вектора a не превышают 1, то в этой линейной комбинации присутствуют только векторы из множества S. ■

Индукцией по числу неравенств в системе легко показать, что теорема 6 имеет место и для СЛОН:

$$S = \begin{cases} L_1(x) = a_{11}x_1 + \ldots + a_{1q}x_q \geq 0, \\ L_2(x) = a_{21}x_1 + \ldots + a_{2q}x_q \geq 0, \\ \ldots \quad \ldots \quad \ldots \quad \ldots \ldots \ldots \quad \ldots \quad \ldots \ldots \\ L_p(x) = a_{p1}x_1 + \ldots + a_{pq}x_q \geq 0, \end{cases} \tag{9}$$

где $a_{ij}, \in \{-1, 0, 1\}$, $x_i \in \{0, 1\}$, $i = 1, \ldots, p$, $j = 1, \ldots, q$.

Рассмотрим некоторые применения вышеприведенных алгоритмов.

2. Поиск множеств независимых вершин неориентированного графа

Множество вершин графа называют независимым, если никакие две вершины этого множества не являются смежными. Задача построения максимального независимого множества вершин графа формулируется таким образом: в заданном неориентованном графе $G = (V, E)$ найти независимое множество вершин максимального размера.

Математическая постановка задачи о независимом множестве вершин имеет такую формулировку. Пусть $x_i \in \{0, 1\}$ — булева переменная, которая равна единице, если вершина $i \in V$ включается в независимое множество вершин графа $G = (V, E)$, и равна нулю в противном случае. Чтобы найти число независимости $\alpha(G)$, достаточно найти одно из решений $x = (x_1, x_2, \ldots, x_{|V|})$ во множестве $\{0, 1\}$, для которого

$$\alpha(G) = \max_{x_i \in \{0,1\}} \sum_{i \in V} x_i \tag{10}$$

при ограничениях

$$x_i + x_j \leq 1, (i, j) \in E. \tag{11}$$

Эта задача включает $|V|$ булевых переменных и $|E|$ ограничений в виде линейных неравенств (11). Эти ограничения означают, что когда вершины i и j смежны, то только одна из них будет принадлежать независимому множеству вершин графа G.

Задачу линейного булевого программирования (10) можно решить, рассматривая только ограничения (11). Действительно, решая систему линейных неравенств (11), находим все множества независимых вершин, а потом среди них находим максимальные множества.

2.1. Первый алгоритм поиска независимых множеств вершин. Простейший алгоритм вычисления всех независимых множеств графа строится традиционным способом: система неравенств (11) трансформируется в систему уравнений путем введения дополнительных неизвестных, решения которой соответствуют искомым независимым множествам.

Пример 4. Пусть дан граф

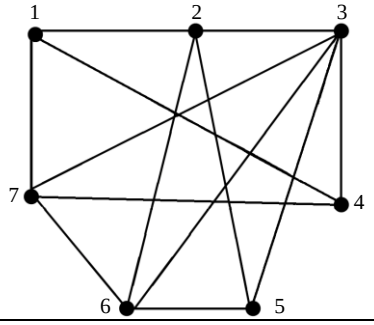

Рис. 1

По системе ограничений (11) графа $G = (V, E)$ строим СЛОУ путем введения дополнительных переменных.

$$S = \begin{cases} 1100000\,|\,1000000000000 - 1 = 0 \\ 1001000\,|\,0100000000000 - 1 = 0 \\ 1000001\,|\,0010000000000 - 1 = 0 \\ 0110000\,|\,0001000000000 - 1 = 0 \\ 0100100\,|\,0000100000000 - 1 = 0 \\ 0100010\,|\,0000010000000 - 1 = 0 \\ 0011000\,|\,0000001000000 - 1 = 0 \\ 0010100\,|\,0000000100000 - 1 = 0 \\ 0010010\,|\,0000000010000 - 1 = 0 \\ 0010001\,|\,0000000001000 - 1 = 0 \\ 0001001\,|\,0000000000100 - 1 = 0 \\ 0000110\,|\,0000000000010 - 1 = 0 \\ 0000011\,|\,0000000000001 - 1 = 0 \end{cases}$$

Решениями данной СЛОУ есть такие векторы:

$$s_1 = (1, 0, 1, 0, 0, 0, 0),\ s_2 = (1, 0, 0, 0, 0, 1, 0),\ s_3 = (1, 0, 0, 0, 1, 0, 0),$$

$$s_4 = (0, 1, 0, 0, 0, 0, 1),\ s_5 = (0, 1, 0, 1, 0, 0, 0),\ s_6 = (0, 0, 0, 1, 0, 1, 0),$$

$$s_7 = (0, 0, 0, 1, 1, 0, 0),\ s_8 = (0, 0, 0, 0, 1, 0, 1),$$

которым соответствуют такие независимые множества вершин:

ISSN 0572-2691

(1, 3), (1, 6), (1, 5), (2, 7), (2, 4), (4, 6), (4, 5), (5, 7).

Поскольку алгоритм находит базис множества решений, то в таком случае будут порождаться все возможные независимые множества, среди которых необходимо выделить максимальные. ♠

2.2. Второй алгоритм построения множеств независимых вершин графа. Множества независимых вершин графа G можно построить путем решения системы линейных однородных неравенств, которые соответствуют системе ограничений (11). Специфика задачи поиска множеств независимых вершин неориентированного графа дает СЛОУ, которая удовлетворяет условиям теоремы 6. Действительно, в каждом неравенстве системы S только два коэффициента положительные и единственный отрицательный коэффициент. А это означает, что при вычислении значений на решениях предыдущих неравенств эти значения будут принадлежать множеству $\{-1, 0, 1\}$.

Пример 5. Возьмем простой пример графа, а именно граф C_5, т.е. цикл, который имеет пять вершин.

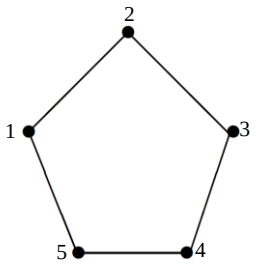

Рис. 2

Для этого графа СЛОН приобретает следующий вид:

$$S = \begin{cases} 1\ 1\ 0\ 0\ 0\ -1 \le 0, \\ 1\ 0\ 0\ 0\ 1\ -1 \le 0, \\ 0\ 1\ 1\ 0\ 0\ -1 \le 0, \\ 0\ 0\ 1\ 1\ 0\ -1 \le 0, \\ 0\ 0\ 0\ 1\ 1\ -1 \le 0. \end{cases}$$

Решениями первого неравенства есть такие векторы (в последней отделенной колонке приведены значения этих векторов на левой части следующего неравенства):

$$\left|\begin{matrix} 100001 \\ 010001 \\ 001000 \\ 000100 \\ 000010 \\ 000001 \end{matrix}\right| \begin{matrix} 0 \\ -1 \\ 0 \\ 0 \\ 1 \\ -1 \end{matrix} \rightarrow \left|\begin{matrix} 100001 \\ 100001 \\ 010011 \\ 001000 \\ 000100 \\ 000011 \\ 000001 \end{matrix}\right| \begin{matrix} -1 \\ 0 \\ 0 \\ 1 \\ 0 \\ -1 \\ -1 \end{matrix} \rightarrow \left|\begin{matrix} 100001 \\ 101001 \\ 010001 \\ 010011 \\ 000100 \\ 001001 \\ 001011 \\ 000011 \\ 000001 \end{matrix}\right| \begin{matrix} -1 \\ 0 \\ -1 \\ -1 \\ 1 \\ 0 \\ 0 \\ -1 \\ -1 \end{matrix} \rightarrow \left|\begin{matrix} 100001 \\ 100101 \\ 101001 \\ 010001 \\ 010101 \\ 010011 \\ 010111 \\ 001011 \\ 001001 \\ 000111 \\ 000101 \\ 000011 \\ 000001 \end{matrix}\right| \begin{matrix} -1 \\ 0 \\ -1 \\ -1 \\ 0 \\ 0 \\ 1 \\ 0 \\ -1 \\ 1 \\ 0 \\ 0 \\ -1 \end{matrix}.$$

Отбрасывая последнюю координату в найденных векторах и нулевой вектор и изымая те векторы, на которых значения в отделенной колонке положительные, находим такое множество решений:

10000, 01000, 00100, 00010, 00001, 10010, 10100, 01010, 01001, 00101.

Этим решениям соответствуют такие множества независимых вершин:

$$\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{1, 4\}, \{1, 3\}, \{2, 4\}, \{2, 5\}, \{3, 5\}. \spadesuit$$

Введем следующее определение.

Определение. Множество вершин неориентированного графа называется непротиворечивым, если оно не включает смежных вершин. Решение СЛОН, которое соответствует графу, называется непротиворечивым, если множество вершин, соответствующих этому решению, непротиворечиво.

Рассмотрим граф из приведенного выше примера, но СЛОН, которая ему соответствует, имеет вид

$$S = \begin{cases} 1\ 1\ 0\ 0\ 0\ -1 \le 0, \\ 0\ 1\ 1\ 0\ 0\ -1 \le 0, \\ 0\ 0\ 1\ 1\ 0\ -1 \le 0, \\ 0\ 0\ 0\ 1\ 1\ -1 \le 0, \\ 1\ 0\ 0\ 0\ 1\ -1 \le 0. \end{cases}$$

Тогда при построении множества решений системы пятое неравенство выступает «фильтром» полученных ранее решений. Действительно, решения первых четырех неравенств имеют последнюю координату, равную 1. А это значит, что дальше никаких комбинаций решений построить невозможно и при подстановках в остальные неравенства полученных решений из них изымаются противоречивые (решения, на которых значения положительные). Итак, правила построения всех множеств независимых вершин неориентированного графа приобретают такой вид:

- упорядочить неравенства системы так, чтобы эти неравенства покрывали все единичные элементы (каждое следующее неравенство включает столбик, который покрывает следующую координату неравенства, которая не покрывалась предыдущими неравенствами);
- если для решения c $L_i(c) = 0$ или $L_i(c) = -1,$ то это решение без каких-либо изменений переходит ко множеству решений неравенства $L_{i+1}(x) \le 0;$
- если для решения c $L_i(c) = -1,$ то это решение комбинируется с решением $y,$ на котором $L_i(y) = 1,$ результат комбинирования, если он непротиворечив, добавляется ко множеству решений неравенства $L_{i+1}(x) = 0;$
- решения c $L_i(c) = 1$ изымаются из дальнейшего рассмотрения.

Если обозначим модифицированный таким образом TSS-алгоритм как $MTSS,$ то для него имеет место следующая теорема.

Теорема 7. Алгоритм $MTSS$ находит все непротиворечивые множества независимых вершин неориентированного графа $G(V, E)$.

Доказательство. Справедливость теоремы следует из того, что при построении множества решений СЛОН все непротиворечивые решения всех неравенств СЛОН сохраняются. Это очевидным образом следует из правил комбинирования при построении множеств решений. ■

 ISSN 0572-2691

К сказанному добавим, что для графа Петерсена алгоритм находит 75 независимых множеств, среди которых пять максимальных независимых множеств размера 4, на графе — шахматной доски размером 4×4

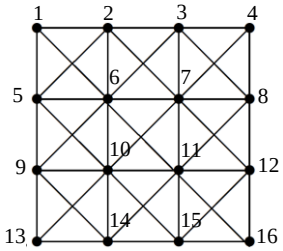

Рис. 3

алгоритм *MTSS* находит все 313 множеств независимых вершин, среди которых 79 множеств максимального размера 4. Соответствующая система неравенств имеет 42 неравенства и 17 неизвестных:

$$S =$$

$$= \begin{cases} L_1 = 1100000000000000-1\le 0, & L_{22} = 0000001001000000-1\le 0, \\ L_2 = 1000100000000000-1\le 0, & L_{23} = 0000001000100000-1\le 0, \\ L_3 = 1000010000000000-1\le 0, & L_{24} = 0000001000010000-1\le 0, \\ L_4 = 0110000000000000-1\le 0, & L_{25} = 0000000100100000-1\le 0, \\ L_5 = 0100100000000000-1\le 0, & L_{26} = 0000000100010000-1\le 0, \\ L_6 = 0100010000000000-1\le 0, & L_{27} = 0000000011000000-1\le 0, \\ L_7 = 0100001000000000-1\le 0, & L_{28} = 0000000010001000-1\le 0, \\ L_8 = 0011000000000000-1\le 0, & L_{29} = 0000000010000100-1\le 0, \\ L_9 = 0010010000000000-1\le 0, & L_{30} = 0000000001100000-1\le 0, \\ L_{10} = 0010001000000000-1\le 0, & L_{31} = 0000000001001000-1\le 0, \\ L_{11} = 0010000100000000-1\le 0, & L_{32} = 0000000001000100-1\le 0, \\ L_{12} = 0001001000000000-1\le 0, & L_{33} = 0000000001000010-1\le 0, \\ L_{13} = 0001000100000000-1\le 0, & L_{34} = 0000000000110000-1\le 0, \\ L_{14} = 0000110000000000-1\le 0, & L_{35} = 0000000000100100-1\le 0, \\ L_{15} = 0000100010000000-1\le 0, & L_{36} = 0000000000100010-1\le 0, \\ L_{16} = 0000100001000000-1\le 0, & L_{37} = 0000000000100001-1\le 0, \\ L_{17} = 0000011000000000-1\le 0, & L_{38} = 0000000000010010-1\le 0, \\ L_{18} = 0000010010000000-1\le 0, & L_{39} = 0000000000010001-1\le 0, \\ L_{19} = 0000010001000000-1\le 0, & L_{40} = 0000000000001100-1\le 0, \\ L_{20} = 0000010000100000-1\le 0, & L_{41} = 0000000000000110-1\le 0, \\ L_{21} = 0000001100000000-1\le 0, & L_{42} = 0000000000000011-1\le 0. \end{cases}$$

Для графа шахматной доски 5×5 система состоит из 72 неравенств и 26 неизвестных, с которой алгоритм находит 6426 независимых множества, среди которых единственное максимальное независимое множество размера 9.

Результаты экспериментов. В табл. 1 приведены временные характеристики работы второго алгоритма (на основе реализации) (в среде macOS, 1.6 GHz Intel Core i5, 8 GB 1600 MHz DDR3, C++).

Таблица 1

Граф	Размерность ограничений	Время вычислений
Шахматная доска 4×4	42×17	7,1 мс
Шахматная доска 5×5	72×26	222,5 мс
Шахматная доска 6×6	110×37	14,5 с
Петерсена, $G(5, 2)$	15×11	2 мс
$G(10, 2)$	30×20	155,1 мс
$G(12, 3)$	36×24	711,3 мс

$G(m, n)$ в таблице обозначает обобщенный граф Петерсена соответствующего размера [4].

2.3. Модификация второго алгоритма. Эффективность алгоритма построения множеств независимых вершин можно улучшить, если после построения решений первого неравенства пометить их координаты, которые не могут принимать участия в комбинациях. Помеченные координаты означают, что соответствующие им вершины смежные с вершинами этого решения. Такие отметки позволяют эффективно выполнять проверку противоречивости решений. Рассмотрим пример.

Пример 6. Для графа C_5 из предыдущего примера с пометками + построение множеств независимых вершин приобретает вид:

$$\left|\begin{array}{cccccc|c} 1 & + & 0 & 0 & + & 1 & 0 \\ + & 1 & + & 0 & 0 & 1 & -1 \\ 0 & + & 1 & + & 0 & 0 & 0 \\ 0 & 0 & + & 1 & + & 0 & 0 \\ + & 0 & 0 & + & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{array}\right| \rightarrow \left|\begin{array}{cccccc|c} 1 & + & 0 & 0 & + & 1 & -1 \\ + & 1 & + & 0 & 0 & 1 & 0 \\ + & 1 & + & + & 1 & 1 & 0 \\ 0 & + & 1 & + & 0 & 0 & 1 \\ 0 & 0 & + & 1 & + & 0 & 0 \\ + & 0 & 0 & + & 1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{array}\right| (2,5) \rightarrow$$

$$\rightarrow \left|\begin{array}{cccccc|c} 1 & + & 0 & 0 & + & 1 & -1 \\ 1 & + & 1 & + & + & 1 & 0 \\ + & 1 & + & 0 & 0 & 1 & -1 \\ 0 & 0 & + & 1 & + & 0 & 1 \\ 0 & + & 1 & + & 0 & 1 & 0 \\ + & + & 1 & + & 1 & 1 & 0 \\ + & 0 & 0 & + & 1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{array}\right| \begin{array}{c} \\ (1,3) \\ \\ \\ \\ (3,5) \\ \\ \end{array} \rightarrow \left|\begin{array}{cccccc|c} 1 & + & 0 & 0 & + & 1 & -1 \\ 1 & + & + & 1 & + & 1 & 0 \\ + & 1 & + & 0 & 0 & 1 & -1 \\ + & 1 & + & 1 & + & 1 & 0 \\ 0 & + & 1 & + & 0 & 1 & -1 \\ 0 & 0 & + & 1 & + & 1 & 0 \\ + & 0 & 0 & + & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{array}\right| \begin{array}{c} \\ (1,4) \\ \\ (2,4) \\ \\ \\ \\ \end{array} .$$

Как видим, в процессе вычисления происходит существенная экономия по сравнению с предыдущим способом. ♠

Результаты экспериментов. В табл. 2 приведены временные характеристики работы второго алгоритма (на основе реализации [3]) (в среде macOS, 1.6 GHz Intel Core i5, 8 GB 1600 MHz DDR3, C++).

Таблица 2

Граф	Размерность ограничений	Время вычислений
Шахматная доска 4×4	72×26	6,9 мс
Шахматная доска 5×5	110×37	168,8 мс
Шахматная доска 6×6	15×11	4,6 с
Петерсена, $G(5, 2)$	30×20	2 мс
$G(10, 2)$	36×24	139,5 мс
$G(12, 3)$	72×26	641,8 мс

ISSN 0572-2691

Как следует из приведенных временных значений, данный алгоритм существенно проигрывает в скорости алгоритму работы [5]. Но алгоритм работы [5] находит только одно наибольшее независимое множество вершин графа, в то время как данная модификация TSS находит все независимые множества вершин графа.

3. Поиск дедлоков и ловушек в сетях Петри

Рассмотрим метод анализа живучести сети Петри (СП), который основывается на применении диофантовых ограничений. Этот метод использует понятие дедлока и ловушки, в терминах которых формулируются соответствующие утверждения о живучести СП [6]. Множество мест Q СП называется дедлоком (ловушкой), если в процессе функционирования СП это множество отдает больше (меньше) фишек, чем получает (отдает). Итак, если произвольный переход имеет выходное место, которое принадлежит дедлоку Q, то в Q должно включаться и его входное место. А для ловушки, наоборот, если произвольный переход имеет входное место, которое относится к ловушке Q, то в Q должно включаться и некоторое его высходное место. Из этого определения следует, что когда дедлок Q является пустым множеством, то он всегда остается пустым множеством в процессе функционирования СП. Для ловушки ситуация противоположная: если хотя бы одно из ее мест получило фишку, то ловушка постоянно остается непустой в процессе функционирования СП. Дедлоки и ловушки могут быть найдены в СП с помощью решения системы логических уравнений, описывающих эти свойства. По системе логических зависимостей строится СЛОН следующим образом: если логическая переменная входит в зависимость со знаком отрицания, то ей соответствует коэффициент –1, без отрицания — коэффициент 1, а при ее отсутствии в зависимости ей соответствует коэффициент 0. Для иллюстрации рассмотрим следующий пример.

Пример 7. Для СП, моделирующей конкурентный вариант алгоритма взаимного исключения, имеем такие множества:

$$^\bullet b_1 = \{Pend_1, Key\}; \quad ^\bullet b_2 = \{Pend_2, Key\}; \quad ^\bullet c_1 = \{Cr_1\};$$

$$^\bullet c_2 = \{Cr_2\}; \quad ^\bullet a_1 = \{quiet_1\}; \quad ^\bullet a_2 = \{quiet_2\};$$

$$b_1^\bullet = \{Cr_1\}; \quad b_2^\bullet = \{Cr_2\}; \quad c_1^\bullet = \{Key, quiet_1\};$$

$$c_2^\bullet = \{Key, quiet_2\}; \quad a_1^\bullet = \{Pend_1\}; \quad a_2^\bullet = \{Pend_2\}.$$

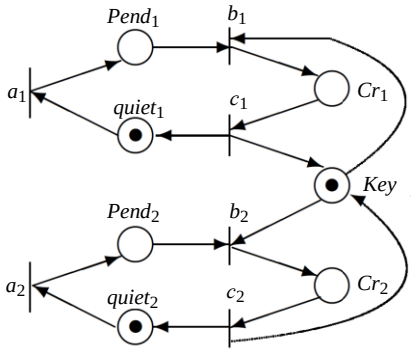

Рис. 4

Отсюда получаем такие логические зависимости для поиска дедлоков в данной СП:

$$\begin{array}{lll}
\text{если} & Key \in S, & \text{то } Cr_1 \wedge Cr_2 \in S;\\
\text{если} & Cr_1 \in S, & \text{то } Pend_1 \vee Key \in S;\\
\text{если} & Cr_2 \in S, & \text{то } Pend_2 \vee Key \in S;\\
\text{если} & Pend_1 \in S, & \text{то } quiet_1 \in S;\\
\text{если} & Pend_2 \in S, & \text{то } quiet_2 \in S;\\
\text{если} & quiet_1 \in S, & \text{то } Cr_1 \in S;\\
\text{если} & quiet_2 \in S, & \text{то } Cr_2 \in S.
\end{array}$$

Далее, условия «если ..., то ...» записываем в виде эквивалентной системы логических формул следующего вида:

$$\begin{array}{ll}
\neg Key \vee (Cr_1 \wedge Cr_2), & \neg Cr_1 \vee Key \vee Pend_1,\\
\neg Cr_2 \vee Key \vee Pend_2, & \neg Pend_1 \vee quiet_1,\\
\neg Pend_2 \vee quiet_2, & \neg quiet_1 \vee Cr_1,\\
\neg quiet_2 \vee Cr_2. &
\end{array}$$

Записывая формулы в виде СЛОН с коэффициентами из множества $\{-1, 0, 1\}$, получаем такую систему:

$$A \cdot x = \left\{ \begin{array}{rrrrrrrl}
-1 & 1 & 0 & 0 & 0 & 0 & 0 \ge 0,\\
-1 & 0 & 1 & 0 & 0 & 0 & 0 \ge 0,\\
1 & -1 & 0 & 1 & 0 & 0 & 0 \ge 0,\\
1 & 0 & -1 & 0 & 0 & 1 & 0 \ge 0,\\
0 & 0 & 0 & -1 & 1 & 0 & 0 \ge 0,\\
0 & 0 & 0 & 0 & 0 & -1 & 1 \ge 0,\\
0 & 1 & 0 & 0 & -1 & 0 & 0 \ge 0,\\
0 & 0 & 1 & 0 & 0 & 0 & -1 \ge 0.\\
Key & Cr_1 & Cr_2 & Pend_1 & quiet_1 & Pend_2 & quiet2
\end{array} \right.$$

Решив данную СЛОН во множестве $\{0, 1\}$, получаем такие векторы:

$$v_1 = (1,1,1,1,1,1,1),\ v_2 = (1,1,1,1,1,0,1),\ v_3 = (1,1,1,1,1,0,0),$$

$$v_4 = (1,1,1,0,1,1,1),\ v_5 = (1,1,1,0,1,0,1),\ v_6 = (1,1,1,0,1,0,0),$$

$$v_7 = (1,1,1,0,0,1,1),\ v_8 = (1,1,1,0,0,0,1),\ v_9 = (1,1,1,0,0,0,0),$$

$$s_{10} = (0,1,0,1,1,0,0),\ s_{11} = (0,0,1,0,0,1,1).$$

Одним из решений данной СЛОН есть, например, векторы

$$x = (1,1,1,1,1,1,1),\ x' = (1,1,1,0,0,0,0).$$

Это означает, что все места данной СП (первое решение) составляют дедлок, как и подмножество мест (второе решение) (Key, Cr_1, Cr_2), причем первое множество не является минимальным дедлоком, а второе множество — минимальный дедлок. ♠

ISSN 0572-2691

Результаты экспериментов. В табл. 3 приведены временные характеристики работы алгоритма поиска дедлоков и ловушек в СП (в среде macOS, 1.6 GHz Intel Core i5, 8 GB 1600 MHz DDR3, C++).

Таблица 3

Количество переменных	Количество формул	Размерность ограничений	Время вычислений
7	8	7×8	0,3 мс
20	5	5×20	1,2 мс
20	10	10×20	7,3 с
100	10	10×100	0,7 с
100	20	20×100	2,1 с

4. Анализ множества дизъюнктов

Рассмотрим одну из модификаций TSS-алгоритма для решения задачи проверки противоречивости множества дизъюнктов в исчислении высказываний. Эта модификация хороша тем, что ее применение позволяет не только проверять противоречивость, а и определять минимальные противоречивые подмножества дизъюнктов. Построим по множеству дизъюнктов S СЛОУ (как это делалось для ловушек и дедлоков) $A_S^{\mathrm{T}} x = 0$ и применим к ней такой алгоритм:

АПС-TSS (S)
Вход. Множество дизъюнктов S.
Выход. ВЫПОЛНИМОЕ или ПРОТЕВОРЕЧИВОЕ.
Метод.
begin
1. Построить A_S^{T} для S;
2. Если в A_S^{T} есть строки, включающие только 0 и 1 или только 0 и – 1, то
(Удалить в A_S^{T} все столбики, которые включают ненулевые элементы этих строк);
3. i:=1;
4. Вычислить TSS для $L_i(x) = 0$;
5. Для всех $j = i + 1$ до p выполнить
Для всех $e \in$ TSS найти значение $L_j(e)$
конец
6. Удалить те $e \in$ TSS, что порождают тавтологии;
(то есть те векторы, для которых значение $L_j(e) = 0$, причем ненулевые координаты e попадают на ненулевые коэффициенты $L_j(x)$)
7. Если TSS $= \varnothing$, то (ВЫПОЛНИМОЕ; на 9);
8. $i = i + 1$; Если $i < p$ то на 4 иначе
Если TSS $= \varnothing$ то ВЫПОЛНИМОЕ иначе ПРОТЕВОРЕЧИВОЕ;
9. СТОП
end

Обоснованием данного алгоритма является следующая теорема.

Теорема 8. Множество дизъюнктов S противоречивое тогда и только тогда, когда СЛОУ $A_S^{\mathrm{T}} x = 0$ имеет непустую TSS, построенную алгоритмом АПС-TSS.

Доказательство. ($\Rightarrow$) Если СЛОУ $A_S^{\mathrm{T}} x = 0$ имеет хотя бы одно ненулевое решение, то на основании построения становится очевидным тот факт, что дизъюнкты, которые соответствуют ненулевым координатам решения, составляют минимальное противоречивое подмножество дизъюнктов. А это значит, что и все множество S тоже будет противоречивым.

($\Leftarrow$) Пусть S противоречивая и $S' = \{D_1, D_2, \ldots, D_k\} \subseteq S$ — ее минимальное противоречивое подмножество, где $k \le m$, $m = |S|$. Среди D_i, $i = 1, 2, \ldots, k$, должна существовать хотя бы одна пара дизъюнктов, резольвента которых нетавтологична. В противном случае получаем противоречие с минимальностью множества S'. Обозначим эту резольвенту D_{js}. Тогда в силу тех же причин, во множестве $(S' \setminus \{D_j\}) \cup \{D_{js}\}$ должна существовать пара дизъюнктов, резольвента которых нетавтологична и т.д. Действуя таким образом дальше, в конце придем к пустому дизъюнкту 0 и получим вектор c, j-я координата которого равна количеству применений правила резолюции, в которых принимал участие дизъюнкт D_j.

А такой вектор будет решением СЛОУ $A_S^{\mathrm{T}} x = 0$. ■

Пример 8. 1. Пусть A_S^{T}

$$A_S^{\mathrm{T}} = \begin{cases} 1 & -1 & 1 & 0 & 1 & -1 & = & 0, \\ -1 & 1 & 1 & -1 & 0 & 1 & = & 0, \\ 1 & 1 & -1 & -1 & 1 & 0 & = & 0, \\ 0 & 0 & -1 & 1 & 1 & -1 & = & 0. \end{cases}$$

соответствует такому множеству дизъюнктов:

$$S = \{p \vee \neg q \vee r,\ \neg p \vee q \vee r,\ p \vee q \vee \neg r \vee \neg u,\ \neg q \vee \neg r \vee u,\ p \vee r \vee u,\ \neg p \vee q \vee \neg u\}.$$

Построение TSS для первого уравнения дает такое множество решений (во 2, 3 и 4 столбиках показаны значения соответственно 2, 3 и 4 уравнений, если подставить в них решения первого уравнения).

	L_2	L_3	L_4
000100	–1	–1	1
110000	0	–	–
011000	2	0	–
010010	1	2	1
100001	0	–	–
001001	2	–1	–2
000011	1	1	0

После чистки получаем такие векторы.

	L_2	L_3	L_4
000100	–1	–1	1
010010	1	2	1
001001	2	–1	–2

Комбинирование по значениям второго столбика порождает следующие векторы.

	L_3	L_4
010110	1	2
001201	–3	0

После чистки получаем единственный вектор.

	L_3	L_4
010110	1	2

ISSN 0572-2691

Дальнейшие комбинации невозможны, поскольку комбинировать не с чем. Итак, начальная СЛОУ не имеет решений (несовместима) и множество дизъюнктов S непротиворечиво.

2. Рассмотрим пример, взятый из учебника по операционным системам и описание которого можно найти в [7].

Есть ли формулы $P \wedge \neg B$ и $I \wedge \neg N$ последствиями нижеследующих формул?

$A \to P$	$\neg A \vee P$
$(P \to \neg B) \to D$	$\neg P \vee B \vee D$
$D \to (S \wedge M \wedge H)$	$\neg D \vee S$
	$\neg D \wedge M$
	$\neg D \wedge H$
$H \wedge N \to R$	$\neg H \vee \neg N \vee R$
$H \wedge \neg R \to I$	$\neg H \vee R \vee I$
$A \wedge \neg B \wedge \neg R$	A
	$\neg B$
	$\neg R$

Добавляем к данным дизъюнктам (выписанных справа в таблице) отрицание формул $P \wedge \neg B$ и $I \wedge \neg N$, которые после приведения принимают вид, который показан в последних двух строках нижеследующей таблицы.

$A \to P$	1) $\neg A \vee P$
$(P \to \neg B) \to D$	2) $\neg P \vee B \vee D$
$D \to (S \wedge M \wedge H)$	3) $\neg D \vee S$
	4) $\neg D \wedge M$
	5) $\neg D \wedge H$
$H \wedge N \to R$	6) $\neg H \vee \neg N \vee R$
$H \wedge \neg R \to I$	7) $\neg H \vee R \vee I$
$A \wedge \neg B \wedge \neg R$	8) A
	9) $\neg B$
	10) $\neg R$
	11) $\neg P \vee B$
	12) $\neg I \vee N$

Из этих зависимостей получаем такую систему логических уравнений:

$$A^{\mathrm{T}} x = \begin{cases} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & \\ A & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & = 0, \\ B & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & = 0, \\ D & 0 & 1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & = 0, \\ P & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & = 0, \\ S & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & = 0, \\ M & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & = 0, \\ H & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & = 0, \\ N & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & = 0, \\ R & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & -1 & 0 & 0 & = 0, \\ I & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & = 0. \end{cases}$$

Применяя алгоритм АПС-TSS к полученному множеству дизъюнктов, находим два решения системы $A^T x = 0$: $x_1 = (1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0)$, ненулевым координатам которого соответствует минимальное противоречивое подмножество дизъюнктов $\neg A \vee P,\ A,\ \neg B,\ \neg P \vee B$, и $x_2 = (2, 2, 0,\ 0, 2, 1, 1, 2, 2, 2, 0, 1)$, ненулевым координатам которого соответствует минимальное противоречивое множество дизъюнктов $\neg A \vee P,\ A,\ \neg B,\ \neg P \vee B\ \vee D,\ \neg D \vee H,\ \neg H \vee \neg N \vee R,\ \neg H \vee\ R \vee I,\ \neg R, \neg I \vee N$. В правильности этих решений можно убедиться, применив правило резолюций для вывода пустого дизъюнкта.

Итак, поскольку отрицание и первой, и второй формул, что проверялись, входит в соответствующие противоречивые подмножества, то сами формулы являются логическими следствиями данного множества дизъюнктов. Кроме того, множества формул, в которые входят эти следствия, — наименьшие подмножества, из которых резолюционным методом выводятся эти следствия. ♠

Результаты экспериментов. В табл. 4 приведены временные характеристики работы алгоритма анализа множества дизъюнктов (в среде macOS, 1.6 GHz Intel Core i5, 8 GB 1600 MHz DDR3, C++).

Таблица 4

Количество переменных	Количество дизъюнктов	Размерность ограничений	Время вычислений
4	6	7×8	0,2 мс
20	5	5×20	0,6 мс
50	20	10×20	8,2 мс
100	50	10×100	48,8 мс
100	20	20×100	10,1 с

Заключение

В настоящей работе рассмотрены общий алгоритм решения СЛОУ и СЛОН в области $\{0, 1\}$ и отдельный класс СЛОУ и СЛОН, коэффициенты которых принадлежат множеству $\{-1, 0, 1\}$. Специфика такого типа ограничений дает возможность построить алгоритмы вычисления всех множеств независимых вершин графа, алгоритмы поиска дедлоков и ловушек в СП и анализа множества дизъюнктов на противоречивость/непротиворечивость.

С.Л. Кривий, В.Т. Антонюк

АЛГОРИТМИ РОЗВ'ЯЗАННЯ СИСТЕМ ЛІНІЙНИХ ОБМЕЖЕНЬ З ЦІЛИМИ КОЕФІЦІЄНТАМИ У МНОЖИНІ {0, 1}

Розглянуто базові поняття систем лінійних однорідних і неоднорідних рівнянь і нерівностей в області {0, 1}, до розв'язання яких застосовується TSS-алгоритм, та властивості цього алгоритму. Описано процедури чистки множин розв'язків та визначено лінійно залежні рівняння системи при роботі TSS-алгоритму. На основі базових понять і властивостей пропонується достатньо економна відносно пам'яті модифікація TSS-алгоритму для чисельного розв'язання систем лінійних однорідних рівнянь і нерівностей з цілими коефіцієнтами в області {0, 1}. Наводиться опис запропонованого алгоритму за допомогою псевдокоду і оцінка часової складності запропонованого алгоритму. Розглянуто алгоритми розв'язання окремого класу систем лінійних однорідних рівнянь і нерівностей, коефіцієнти яких належать множині {– 1, 0, 1}. Наведено ряд теорем, що доводять правильність пропонованих алгоритмів. Описано їх застосування до розв'язання наступних задач: пошуку множин незалежних вершин неорієнтованого графа; пошуку дедлоків і пасток в мережі Петрі; аналізу множини диз'юнктів на суперечність/несуперечність. Для задачі пошуку множин незалежних вершин неорієнтованого графа наведено детальний опис зведення задачі до системи лінійних однорідних нерівностей, запропоновано

ISSN 0572-2691

два алгоритми розв'язання, а також модифікація другого алгоритму. Наведено приклади з детальним поясненням виконання кожного з алгоритмів, а також описано їх часові характеристики роботи. Для задач пошуку дедлоків і пасток в мережі Петрі запропоновано спосіб зведення до систем лінійних нерівностей з коефіцієнтами в множині {– 1, 0, 1} та розв'язками в множині {0, 1}. Наведено приклад з поясненням розв'язання і часові характеристики роботи запропонованого алгоритму. Алгоритм для аналізу множини диз'юнктів на суперечність представлений у вигляді псевдокоду. Окрім перевірки на суперечність заданої множини диз'юнктів, він дозволяє знаходити мінімальні суперечні підмножини диз'юнктів, якщо вони існують. Робота алгоритму ілюструється на прикладах з часовими характеристиками.

Ключові слова: системи діофантових рівнянь, нерівностей, базис множини розв'язків, незалежні множини вершин, пастки та дедлоки мереж Петрі.

S.L. Kryvyi, V.T. Antonyuk

ALGORITHMS FOR SOLVING THE SYSTEMS OF LINEAR CONSTRAINTS WITH INTEGER COEFFICIENTS IN THE SET {0, 1}

The basic concepts of linear homogeneous and inhomogeneous equations and inequalities in the domain {0, 1}, are considered the properties of the TSS-algorithm, which may be applied in solving those systems are described. The procedures for clearing the sets of solutions and determining the linearly dependent equations of the system during the process of the TSS algorithm are shown. Based on the basic concepts and properties, a memory effective modification of the TSS-algorithm for solving systems of linear homogeneous equations and inequalities with integer coefficients in {0, 1} is offered. A description of the proposed algorithm using pseudocode and an estimate of the time complexity are given. Algorithms for solving a separate class of systems of linear homogeneous equations and inequalities whose coefficients belong to the set {– 1, 0, 1} are considered. A series of theorems that prove the correctness of the proposed algorithms are given. It is described their application to the following problems: finding sets of independent vertices of an undirected graph; finding deadlocks and traps in the Petri net; analysis of a set of clauses for inconsistency. For the task of finding sets of independent vertices of an undirected graph, a detailed description of reducing the problem to a system of linear inequalities is given, two solution algorithms are proposed, as well as a modification of the second algorithm. Examples are presented with a detailed explanation of the solution via each of the algorithms and their time characteristics of work are described. For problems of finding deadlocks and traps in the Petri net, a method for reducing it to systems of linear inequalities with coefficients in the set {– 1, 0, 1} and solutions in the set {0, 1} is proposed. An example with the solution explanation and time characteristics of the work of the offered algorithm is described. The algorithm for analyzing the set of clauses for inconsistency is presented in a pseudo-code form. In addition to checking for the inconsistency of a given set of clauses, it allows you to find the minimal inconsistent subsets of clauses if they exist. The operation of the algorithm is illustrated with examples and time characteristics of the algorithm are described.

Keywords: systems of Diophantine equations, inequalities, basis of the set of solutions, independent sets of vertices, traps and deadlocks of Petri nets.

1. Кривий С.Л. Лінійні діофантові обмеження та їх застосування. Чернівці; Київ : Букрек. 2015. 224 с.
2. Gross J.L., Yellen J. Handbook of graph theory. CRC Pres. 2004. **1**. chap. 5 (Independent Sets and Cliques). P. 389–402.
3. Кривий С.Л., Антонюк В.Т. Реалізація алгоритму розв'язання системи лінійних діофантових рівнянь в кільці лишків. *Управляющие системы и машины.* 2017. № 6. С. 55–64.
4. Википедия. Обобщенный граф Петерсена.
5. Кристофидес Н. Теория графов. Алгоритмический подход. М: Мир, 1978. 432 с.
6. Murata T. Petri nets: Properties, analysis and applications. *Proc. of the IEEE*. 1989. **77**, N 4. P. 541–580.
7. Ross K., Wright C.R.B. Discrete mathematics. 4-th ed. Upper Saddle River. NY: Prentice-Hall. 1999. 761 p.

Получено 27.03.2019