УДК 519.987

ЭФФЕКТИВНЫЕ МЕТОДЫ НАХОЖДЕНИЯ КРАТЧАЙШИХ РЕШЕНИЙ СИСТЕМ ЛИНЕЙНЫХ ЛОГИЧЕСКИХ УРАВНЕНИЙ[1]

А.Д. Закревский

Объединенный институт проблем информатики, г. Минск

Предложены эффективные методы, существенно сокращающие объем перебора, производимого при поиске кратчайшего решения. Приведены результаты их программной реализации, дано оценивание конкурентоспособности разработанных программ и указаны области их применимости.

ВВЕДЕНИЕ

С задачей нахождения кратчайших решений систем линейных логических уравнений (СЛЛУ) автор встретился при разработке методов синтеза логических двухуровневых AND/EXOR-схем, обладающих некоторыми преимуществами в сравнении с традиционными AND/OR-схемами [1–3 и др.]. Например, их легче тестировать [4–6], и они оказываются более компактными при реализации симметрических булевых функций, типичных для арифметики [7]. Для решения задач их проектирования оказалось возможным привлечь хорошо развитую теорию линейных векторных пространств [8, 9]. Не менее важные приложения данная задача имеет в такой бурно развивающейся области, как защита информации [10].

В абстрактном виде задача формулируется следующим образом.

Пусть задана система из m линейных логических уравнений с n переменными:

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}, \qquad \bigoplus_{j=1}^{n} a_i^j x_j = y_i,$$

где $\boldsymbol{A}$ – булева $m \times n$ матрица коэффициентов, $\boldsymbol{x} = (x_1, x_2, ..., x_n)$ – булев вектор неизвестных и $\boldsymbol{y} = (y_1, y_2, ..., y_m)$ – булев вектор свободных членов. Решением СЛЛУ называется такой набор значений неизвестных, который обращает каждое уравнение в тождество. В таком случае покомпонентная сумма по модулю два (называемая ниже просто суммой) тех столбцов матрицы $\boldsymbol{A}$, соответствующие которым переменные принимают в решении значение 1, будет равна столбцу $\boldsymbol{y}$.

Положим, что СЛЛУ имеет несколько решений (тогда система называется *неопределенной*, при этом обычно $n > m$). Задача заключается в выборе из них *кратчайшего решения*, т. е. такого, которое представлено значением вектора $\boldsymbol{x}$ с минимальным числом единиц.

В принципе эту задачу можно решить тривиальным алгоритмом, который перебирает сочетания «из n по k» столбцов матрицы $\boldsymbol{A}$, в порядке возрастания k и начиная с $k = 1$, подсчитывает их сумму и сравнивает с вектором $\boldsymbol{y}$. Достаточно очевидно, что как только получаемая сумма оказывается равной $\boldsymbol{y}$, рассматриваемая комбинация столбцов может быть принята за искомое решение. К сожалению, данный алгоритм слишком трудоемок, поскольку число анализируемых комбинаций столбцов очень быстро растет с ростом числа неизвестных n.

В следующих разделах излагаются более эффективные комбинаторные методы решения данной задачи, ориентированные на компьютерную реализацию.

ЛЕСТНИЧНЫЙ АЛГОРИТМ

Оригинальный алгоритм поиска кратчайшего решения СЛЛУ был предложен при разработке ме-

[1] Работа доложена на конференции «Теория и практика логического управления», посвященной памяти М.А. Гаврилова (Москва, 10 ноября 2003 г.).

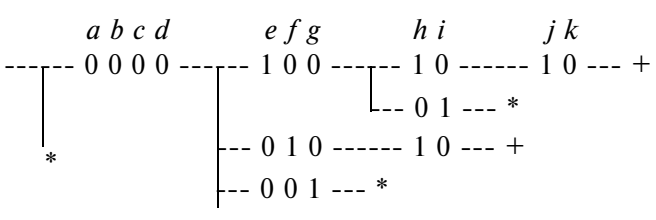
тодов реализации частичных булевых функций полиномами Жегалкина, что оказалось полезным при синтезе логических схем в базисе элементов AND (логическое И) и EXOR (сумма по модулю два) [11, 12].

Этот алгоритм назван лестничным, и его суть заключается в следующем. Путем соответствующего переупорядочения строк и столбцов матрица коэффициентов $\boldsymbol{A}$ приводится к ступенчатой форме, иллюстрируемой приводимым ниже примером, в котором $\boldsymbol{x} = (a, b, c, d, e, f, g, h, i, j, k, l)$. При этом каждая последующая строка матрицы $\boldsymbol{A}$ не поглощается покомпонентной дизъюнкцией предыдущих (это оказывается возможным, когда конструируется компактный полином Жегалкина), а оставшиеся непоглощенными единичные элементы образуют в строке своеобразную ступеньку, ширина которой равна числу этих единиц:

$$\boldsymbol{A} = \begin{array}{c} a\ b\ c\ d\ e\ f\ g\ h\ i\ j\ k\ l \\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\ 1\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0 \\ 1\ 1\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0 \\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 0 \\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 1 \end{array}, \quad \boldsymbol{y} = \begin{array}{c} y \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{array}$$

Матрица коэффициентов $\boldsymbol{A}$ с присоединенным к ней вектор-столбцом свободных членов $\boldsymbol{y}$ образует *расширенную* матрицу системы, или *системную* матрицу $\boldsymbol{S} = (\boldsymbol{A}, \boldsymbol{y})$ – конкатенацию матрицы $\boldsymbol{A}$ и вектора $\boldsymbol{y}$. Каждая ее строка представляет отдельное уравнение. Если $y = 0$, оно будет удовлетворено при включении в решение четного числа столбцов из $\boldsymbol{A}$, отмеченных в данной строке значением 1. Если же $y = 1$, в решение потребуется включить нечетное число таких столбцов. Учитывая это требование, мы организуем поиск по дереву [13], стремясь найти минимальную совокупность столбцов, сумма которых равна вектору $\boldsymbol{y}$. Этот процесс можно интерпретировать как бег по лестнице, в котором спуски (вниз по матрице, с целью решения последующих уравнений) чередуются с подъемами (с целью возможного упрощения решения системы в целом путем испытания альтернативных комбинаций).

Для рассматриваемого примера процесс отображается следующим деревом поиска:



Прокомментируем его. Процесс начинается с анализа первой сверху строки и выбора комбинации значений неизвестных a, b, c, d, перечисленных в верхней ступеньке. Минимизируя число единиц, выбираем комбинацию 0000, удовлетворяющую первому уравнению. Рассматривая последующие строки и ступеньки, выбираем $(e, f, g) = 100$, $(h, i) = 10$, $(j, k) = 10$ и находим таким образом первое решение – тройку столбцов (e, h, j), сумма которых равна вектору $\boldsymbol{y}$. Далее пытаемся найти лучшее решение, содержащее не более двух столбцов. С этой целью идем вверх по лестнице и испытываем комбинацию 01 значений переменных h, i (с отрицательным результатом) и комбинацию 010 значений переменных e, f, g (что приводит затем к нахождению двухстолбцового решения (f, h)). Это окончательное решение, поскольку найти лучшее решение не удается.

Более подробно лестничный алгоритм описан в работе [12], а в статье [14] предложен его ускоренный вариант, основанный на минимизации ширины верхних ступенек, что существенно сокращает объем перебора комбинаций, производимого при поиске оптимального решения.

МЕТОД ГАУССА

Более перспективным оказался метод Гаусса исключения переменных. Он позволяет ограничить поиск решения рассмотрением только таких значений вектора $\boldsymbol{x}$, которые заведомо являются корнями системы, что сокращает число рассматриваемых подмножеств в 2^m раз, в сравнении с тривиальным алгоритмом. Данный метод был предложен Гауссом для решения систем линейных уравнений с действительными переменными [15] и описывается ниже в модификации для нашего случая: переменные – булевы, сумма – по модулю два, произведение – логическое (конъюнкция).

Метод Гаусса основан на следующих положениях. Две системы (а также представляющие их системные матрицы $\boldsymbol{S}_i$ и $\boldsymbol{S}_j$) называются эквивалентными, если множества их корней равны. Матрицы $\boldsymbol{S}_i$ и $\boldsymbol{S}_j$ эквивалентны, если каждая строка одной из них (любой) равна сумме некоторых строк другой матрицы. Любую матрицу $\boldsymbol{S}_i$, эквивалентную данной матрице $\boldsymbol{S}$, можно получить из $\boldsymbol{S}$ некоторой последовательностью операций сложения строк. Эта операция заключается в замене произвольной строки рассматриваемой матрицы суммой этой строки с некоторой другой строкой этой же матрицы. Можно также дублировать некоторые строки и “приводить подобные”, т.е. устранять дубли. Минимальная по числу строк матрица $\boldsymbol{S}_i$, эквивалентная исходной матрице $\boldsymbol{S}$, состоит из ли-

нейно независимых строк – никакая из них не равна сумме некоторых других. Число этих независимых строк называется рангом матрицы $\boldsymbol{S}$ и обозначается через $r(\boldsymbol{S})$.

Метод Гаусса приводит матрицу $\boldsymbol{S}$ к удобной для последующего анализа форме, называемой *канонической*. Делается это (для булевой матрицы) следующим образом.

- Вначале все строки и столбцы матрицы $\boldsymbol{S}$ считаются неотмеченными.
- На очередном шаге выбирается первый из столбцов (за исключением $\boldsymbol{y}$), содержащих единицу в какой-либо из неотмеченных строк, а также первая из неотмеченных строк, содержащих единицу в этом столбце.
- Каждая строка (кроме выбранной), содержащая единицу в выбранном столбце, заменяется ее суммой с выбранной строкой. При получении строки $\mathbf{0}$ (все компоненты равны нулю) она удаляется (так устраняется избыточность в системе). Если же результат суммирования представляется строкой с единственной единицей в правой компоненте (такая строка задает уравнение $0 = 1$, что свидетельствует о противоречивости системы), то алгоритм прекращает работу.
- Выбранные таким образом строка и столбец отмечаются (полужирной единицей на их пересечении), после чего выполняется следующий шаг. Алгоритм прекращает работу, когда все строки или все столбцы становятся отмеченными.

В результате получается каноническая матрица $\boldsymbol{S}^*$, эквивалентная исходной матрице $\boldsymbol{S}$. Совокупность отмеченных столбцов (их число равно рангу матрицы и каждый из них содержит ровно по одной единице) назовем базисом, совокупность остальных – *остатком*. Матрица $\boldsymbol{S}^*$ представляет в компактной форме множество всех корней рассматриваемой системы уравнений, соответствующих различным подмножествам столбцов из остатка.

Например,

$$\boldsymbol{S} = \begin{array}{c} a\ b\ c\ d\ e\ f\ g\ h\ y \\ \hline 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 0 \\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 1 \\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1 \\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ 0 \\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 1 \end{array} \rightarrow \boldsymbol{S}^* = \begin{array}{c} a\ b\ c\ d\ e\ f\ g\ h\ y \\ \hline \mathbf{1}\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1 \\ 0\ 0\ \mathbf{1}\ 0\ 0\ 1\ 0\ 1\ 0 \\ 0\ \mathbf{1}\ 0\ 0\ 0\ 0\ 1\ 1\ 1 \\ 0\ 0\ 0\ \mathbf{1}\ 0\ 1\ 1\ 0\ 0 \\ 0\ 0\ 0\ 0\ \mathbf{1}\ 0\ 0\ 1\ 1 \end{array}$$

Рассматриваемая в этом примере матрица $\boldsymbol{S}$ является неопределенной, $n > m$ и $m = r(\boldsymbol{S})$. Число корней такой матрицы равно 2^{n-m}, в данном случае восьми. В ходе пошагового преобразования матрицы в базис попадают столбцы, соответствующие переменным a, b, c, d, e, а остаток представляется множеством столбцов, соответствующих переменным f, g, h. Каждое из восьми подмножеств этого множества определяет одно из решений, которое легко получить, складывая столбцы подмножества и дополняя их некоторыми столбцами из базиса так, чтобы получить в сумме столбец y. Например, включив в решение переменные f и h и сложив соответствующие им векторы 11010 и 01101 с вектором 10101, мы получим вектор 00010, откуда следует, что решение надо дополнить переменной d. Так находится одно из восьми решений системы, а именно $y = d \oplus f \oplus h$.

Чтобы решить поставленную выше задачу, достаточно перебрать таким образом все 2^{n-m} корней системы и выбрать из них кратчайший. Можно сократить перебор в процессе поиска, если уже найден некоторый корень с весом v. В этом случае достаточно ограничиться поиском лучшего решения, перебирая в остатке подмножества мощностью менее v. Эта идея реализована в описываемом ниже алгоритме.

Допустим, что рассматриваемая система линейных логических уравнений задана введенной выше системной булевой матрицей $\boldsymbol{S}$, причем $n > m$. Тогда кратчайший корень системы можно найти следующим образом [16].

- Приведем матрицу $\boldsymbol{S}$ к канонической форме $\boldsymbol{S}^*$, определив ее базис и остаток. Для этого используется описанный выше алгоритм, попутно проверяющий систему на совместность.
- Будем рассматривать подмножества столбцов из остатка в порядке возрастания их мощности k и взвешивать определяемые ими корни. Вес корня равен количеству выбранных столбцов плюс вес булева вектора (число единиц в нем), представляющего сумму (покомпонентную, по модулю два) этих столбцов и вектора $\boldsymbol{y}$. По ходу будем запоминать решение с минимальным весом v. Прекратим перебор, как только k сравняется с v. В этом случае запомненное решение представит кратчайший корень системы.

Полезно оценить ожидаемый вес $\gamma(m, n)$ кратчайшего корня системы уравнений с параметрами m и n еще до его нахождения – тогда можно будет предсказать и затраты времени на его поиск.

Начнем с нахождения математического ожидания $\alpha(m, n, k)$ числа корней с весом k. Положим, что рассматриваемая система случайна – каждый элемент булевой матрицы $\boldsymbol{S}$ принимает значение 1 с вероятностью 0,5. Тогда вероятность того, что случайно выбранное подмножество столбцов в матрице $\boldsymbol{A}$ окажется корнем, равна 2^{-m} (будучи равна вероятности совпадения двух булевых векторов с m компонентами, один из которых случаен). Отсюда следует

Утверждение 1. $\alpha(m, n, k) = C_n^k 2^{-m}$, где C_n^k – число сочетаний из n по k.

Аналогично, обозначив через $\beta(m, n, k)$ математическое ожидание числа корней, вес которых не превышает k, получим

Утверждение 2. $\beta(m, n, k) = \sum_{i=0}^{k} C_n^i 2^{-m}$.

Искомый вес кратчайшего корня оценим максимальным значением параметра k, при котором $\beta(m, n, k) < 1$.

Утверждение 3. $\gamma(m, n) = k$, где $\beta(m, n, k) < 1 \le \beta(m, n, k+1)$.

Например, в табл. 1 показаны значения величин α и β для системы из 40 уравнений с 70-ю переменными, соответствующие значениям параметра k от 7 до 13. Достаточно очевидно, что вес кратчайшего решения, скорее всего, будет равен 10.

Оценивая объем вычислений при поиске кратчайшего решения, положим, что он пропорционален числу N рассматриваемых подмножеств столбцов остатка. Очевидно, значение N определяется уровнем перебора, который мы обозначим $\delta(m, n)$.

Утверждение 4. $N(m, n) = \sum_{i=0}^{\delta(m,n)} C_{n-m}^i$.

Достаточно очевидно, что объем перебора в сильной степени зависит от мощности остатка – числа столбцов в нем, которое равно разности $n - m$. В случае, когда это число оказывается меньшим веса кратчайшего корня, для гарантированного нахождения последнего приходится перебирать все 2^{n-m} подмножества столбцов в остатке, что приводит к удвоению времени поиска решения при каждом увеличении разности $n - m$ на единицу. Легко подсчитать, что при увеличении разности $n - m$ на 10 время поиска возрастает примерно в тысячу раз.

Как показывают аналитические и экспериментальные (C++, PC COMPAC Presario – процессор Intel Pentium III, 1000 МГц) оценки эффективности рассмотренного метода, область его практического приложения ограничивается условием $n - m < 30$. При этом условии время поиска не превышает одного часа, в то время как с дальнейшим ростом разности $n - m$ расходы времени резко возрастают. Например, при $m = 625$ и $n = 700$ данный алгоритм будет вынужден перебирать и испытывать в поисках кратчайшего решения все 2^{75} комбинаций наборов столбцов из остатка, и расчеты показывают, что на это уйдет три миллиарда четыреста миллионов лет!

Таблица 1

Значения α и β при $m = 40$, $n = 70$ и $k = 7...13$

k	α	β
7	0,001	0,001
8	0,009	0,010
9	0,059	0,069
10	0,361	0,430
11	1,968	2,398
12	9,676	12,074
13	43,170	55,244

РАСПОЗНАВАНИЕ КОРОТКИХ РЕШЕНИЙ

Между тем, кратчайшее решение, вес которого $\gamma(m, n)$ можно заранее оценить по формуле утверждения 3, будет содержать в остатке в среднем $\gamma(m, n) \cdot (n - m)/n$ элементов, откуда следует, что оно, скорее всего, будет найдено на уровне перебора $\delta(m, n) = \lfloor \gamma(m, n) \cdot (n - m)/n \rfloor$ – окаймляющие скобки означают «целое, ближнее снизу». В этом случае его можно *распознать* по весу и прекратить поиск, отказавшись от строгого доказательства оптимальности найденного решения и сэкономив тем самым время. При $m = 625$ и $n = 700$ ожидаемый вес γ кратчайшего решения равен 220, следовательно, уровень перебора $\delta = \lfloor 220 \cdot 75/700 \rfloor = 23$. На этом уровне решение будет найдено за 1 800 000 лет, примерно в 1900 раз быстрее, чем при полном переборе, однако недостаточно быстро с практической точки зрения. На более существенную экономию времени можно рассчитывать, удовлетворившись некоторым приближением к γ сверху.

Практический интерес представляет случай существования *короткого* решения СЛЛУ со случайной матрицей $\boldsymbol{A}$ и вектором $\boldsymbol{y}$, равным сумме некоторых w столбцов этой матрицы (при этом w заметно меньше γ). Например, при $m = 625$, $n = 700$ и $w = 105$ уровень перебора δ будет в среднем равен 11, откуда следует, что затраты времени на нахождение данного решения методом распознавания составят 193 дня. Однако, если повезет, уровень перебора может оказаться и ниже, что приведет к большой экономии времени. Так, при проведении эксперимента на компьютере указанное короткое решение было обнаружено на уровне перебора 10, что соответствует затратам времени в 25 дней.

ДЕКОМПОЗИЦИОННЫЙ МЕТОД НЕПЕРЕСЕКАЮЩИХСЯ ОСТАТКОВ

При больших значениях разности $n - m$ решение рассматриваемой задачи можно существенно ускорить, применяя декомпозиционный метод не-

пересекающихся остатков, понижающий уровень перебора [17–19].

Предположим, что в матрице $\boldsymbol{A}$ можно найти q таких максимальных совокупностей линейно независимых столбцов, которым соответствуют взаимно непересекающиеся остатки. Заметим, что при этом должно выполняться отношение $n \geq q(n - m)$. Построим множество Q, составленное из q соответствующих канонических форм, базисы которых строятся на основе указанных совокупностей (очевидно, что это число ограничено отношением $q \leq n/(n - m)$), и будем искать оптимальное решение в рамках этих форм, последовательно увеличивая уровень перебора.

Утверждение 5. В множестве Q найдется каноническая форма, где решение с весом v можно найти на уровне перебора не выше $\lfloor v/q \rfloor$ – натурального числа, ближнего снизу к частному v/q.

Утверждение 6. Кратчайшее решение системы можно найти путем последовательного рассмотрения остатков канонических форм из множества Q, ограничивая уровень перебора подмножеств в этих остатках величиной $\lfloor (v - 1)/q \rfloor$, где v – вес кратчайшего из решений, найденных к текущему моменту времени.

Эти утверждения образуют основу предлагаемого метода. Обратим внимание на существенное понижение уровня перебора подмножеств в этом методе, в котором реализуется перебор подмножеств во всех остатках сначала на уровне 0, затем на уровне 1, и т. д., пока этот уровень не превысит $\lfloor (v - 1)/q \rfloor$.

Утверждение 7. Число $N_r(m, n)$ подмножеств столбцов, рассматриваемых при методе непересекающихся остатков, определяется следующей формулой:

$$N_r(m, n) = q \sum_{i=0}^{p} C_{n-m}^{i}, \text{ где } p = \lfloor \gamma/q \rfloor.$$

Предложенный метод позволяет существенно сократить время гарантированного нахождения кратчайшего решения, в сравнении с описанным выше методом Гаусса. Для того же примера (m = 625, n = 700) уровень перебора понизится с 220 до 23, как и у метода распознавания, с соответствующим сокращением времени поиска. Однако, в отличие от последнего, при этом гарантируется оптимальность получаемого решения.

Обратим внимание на одно весьма существенное для дальнейших рассуждений обстоятельство. Столбцы матрицы $\boldsymbol{A}$, составляющие кратчайшее решение, распределяются по остаткам канонических форм случайно, т. е. не равномерно, а с некоторой дисперсией, благодаря чему найдется остаток с минимальным числом k вошедших туда столбцов, заметно меньшим, чем $\lfloor \gamma/q \rfloor$ (обозначим через s номер этого остатка). Это позволяет, объединив методы декомпозиции и распознавания, найти кратчайшее решение на уровне перебора k.

Наибольший эффект такое объединение дает в случае существования короткого решения с весом w. Например, при m = 625 и n = 700 число канонических форм q равно 9. При анализе на компьютере примера с данными параметрами и w = 105 оказалось, что k = 7 и именно таким числом элементов короткого решения обладает остаток третьей формы (s = 3). Расчеты по методу, предложенному в работе [20], показывают, что в данном случае решение будет найдено за 7,5 ч.

ОБОБЩЕННЫЙ ДЕКОМПОЗИЦИОННЫЙ МЕТОД

Условие не пересечения остатков канонических форм существенно для метода, гарантирующего оптимальность находимых решений, но не для метода распознавания короткого решения, где более важно понизить уровень перебора. С учетом этого соображения декомпозиционный метод был обобщен следующим образом. На базе различных выборов из матрицы $\boldsymbol{A}$ максимальных множеств линейно независимых столбцов строится несколько (обозначим их число через p) серий канонических форм, причем в каждой из этих серий данное условие не пересечения выполняется. Таким образом, число канонических форм увеличивается в два, три или более раз, что приводит к соответствующему росту времени на их анализ. Однако с большой вероятностью этот расход времени с лихвой компенсируется снижением значения параметра k, определяющего уровень производимого перебора.

В результате возникает выигрыш, хорошо иллюстрируемый на примере той же СЛЛУ с параметрами m = 625, n = 700, обладающей коротким решением (w = 105). Так, в эксперименте на компьютере при p = 2 (две серии, q = 18) решение находится на уровне перебора k = 6 (s = 12) примерно за три часа. А при p = 3 (три серии, q = 27) решение находится на уровне перебора k = 4 (s = 12), всего за 78 с!

ЭКСПЕРИМЕНТЫ

Приведенный выше пример оказался весьма выигрышным для предлагаемых методов нахождения короткого решения СЛЛУ, демонстрируя снижение уровня перебора с 10 до 4, а времени решения – с 25 дней до 78 с.

Таблица 2

Результаты испытания методов нахождения короткого решения случайных СЛЛУ

№	$p = 0$		$p = 1$		$p = 2$		$p = 3$	
	q-s-k	t_0	q-s-k	t_1	q-s-k	t_2	q-s-k	t_3
1	1-1-12	74 100 000	9-2-8	**177 000**	18-13-6	**11 200**	27-13-5	**996**
2	1-1-12	74 100 000	9-3-7	**27 200**	18-3-7	34 900	27-3-7	42 600
3	1-1-10	2 190 000	9-3-7	**27 200**	18-12-6	**10 400**	27-12-4	**78**
4	1-1-12	74 100 000	9-4-6	**3 410**	18-4-6	4 070	27-4-6	4 730
5	1-1-11	13 400 000	9-4-7	**34 900**	18-11-5	**807**	27-13-6	11 800
6	1-1-9	320 000	9-3-7	**27 200**	18-3-7	34 900	27-3-7	42 600
7	1-1-13	373 000 000	9-3-8	**243 000**	18-17-6	**14 300**	27-20-7	175 000
8	1-1-9	320 000	9-5-7	**42 700**	18-5-7	50 400	27-5-7	58 200
9	1-1-13	373 000 000	9-3-8	**243 000**	18-10-7	**89 200**	27-3-8	398 000
10	1-1-12	74 100 000	9-2-6	**1 840**	18-2-6	2 500	27-2-6	3 160
11	1-1-9	320 000	9-1-9	917 000	18-1-9	1 590 000	27-27-7	**229 000**
12	1-1-8	41 600	9-1-8	111 000	18-18-6	**15 100**	27-27-6	22 800
13	1-1-11	13 400 000	9-6-7	**50 400**	18-6-7	58 200	27-23-5	**1 670**
14	1-1-10	2 190 000	9-7-8	**507 000**	18-15-7	**128 000**	27-7-8	662 000
15	1-1-8	41 600	9-3-6	**2 620**	18-3-6	3 280	27-3-6	3 940
16	1-1-9	320 000	9-5-8	375 000	18-11-6	**9 580**	27-16-6	14 200
17	1-1-10	2 190 000	9-2-6	**1 840**	18-2-6	2 500	27-2-6	3 160
18	1-1-15	7 270 000 000	9-2-8	**177 000**	18-15-7	**128 000**	27-11-5	**10 200**
19	1-1-14	1 720 000 000	9-4-9	**2 390 000**	18-13-6	**11 200**	27-13-6	11 800
20	1-1-15	7 270 000 000	9-8-6	**6 560**	18-8-6	7 220	27-8-6	7 880
21	1-1-10	2 190 000	9-7-6	**5 780**	18-7-6	6 430	27-7-6	7 090
22	1-1-9	320 000	9-3-7	**27 200**	18-3-7	34 900	27-22-6	**18 900**
23	1-1-12	74 100 000	9-9-7	**73 700**	18-9-7	81 500	27-20-5	**1 470**
24	1-1-4	3	9-1-4	**8**	18-1-4	15	27-1-4	25
25	1-1-12	74 100 000	9-4-5	**284**	18-4-5	335	27-4-5	388
26	1-1-7	4 740	9-1-7	11 600	18-1-7	19 400	27-1-7	27 100
27	1-1-13	373 000 000	9-8-5	**554**	18-8-5	605	27-8-5	658
28	1-1-11	13 400 000	9-4-6	**3 410**	18-4-6	4 070	27-4-6	4 730
29	1-1-11	13 400 000	9-5-9	**2 880 000**	18-18-6	**15 100**	27-18-7	159 000
30	1-1-13	373 000 000	9-4-7	**34 900**	18-4-7	42 700	27-4-7	50 400

Более объективно эффективность данных методов характеризуется табл. 2, где показаны прогнозируемые результаты решения тридцати случайных СЛЛУ с теми же параметрами ($m = 625$, $n = 700$, $w = 105$) четырьмя методами. Значением $p = 0$ условно отмечен метод Гаусса, значением $p = 1$ – декомпозиционный метод на одной серии канонических форм с непересекающимися остатками, и значениями $p = 2$ и $p = 3$ – обобщенные декомпозиционные методы на двух и трех сериях. Результаты применения этих методов отображаются тройками значений параметров $q - s - k$ и временем t в секундах, затраченным на нахождение решения. Например, тройка 27-12-4 означает, что в процессе решения анализировалось 27 канонических форм и процесс успешно завершился при рассмотрении остатка формы 12 на уровне перебора 4. Полужирным шрифтом отмечены моменты, в которых рассматриваемый метод выигрывает по эффективности у всех предшествующих (представленных столбцами слева).

Поскольку время приведено в секундах, заметим, что час содержит 3600 с, сутки – 86 400 с, год – 31 536 000 с. Так, пример 20 решается методом Гаусса ($p = 0$) за 230 лет, а методом декомпозиции ($p = 1$) – за два часа (в миллион раз быстрее).

ЗАКЛЮЧЕНИЕ

Нахождение кратчайшего решения неопределенной системы линейных логических уравнений является ярким примером трудных комбинаторных задач. Для их успешного решения необходимо не только привлекать компьютеры, но и разрабатывать эффективные комбинаторные методы, учитывающие особенности рассматриваемых задач. Такие методы, предложенные в настоящей статье, на порядки (в тысячи раз и более) могут сокращать время, затрачиваемое на поиск кратчайших решений.

ЛИТЕРАТУРА

1. *Besslich P.W.* Efficient computer method for XOR logic design // IEE Proc. on Computers & Digital Techniques. – 1983. – Vol. 130. – P. 203–206.
2. *Sasao T.* AND-EXOR expressions and their optimization // Logic Synthesis and Optimization / Kluwer Academic Publishers. – 1993. – P. 287–312.
3. *Miller J.F., Thomson P.* Highly efficient exhaustive search algorithm for optimizing canonical Reed-Muller expansions of boolean functions // Int. J. Electronics. – 1994. – Vol. 76. – P. 37–56.
4. *Reddy S. M.* Easily testable realizations for logic functions // IEEE Trans. – 1972. – C-21. – P. 1183–1188.
5. *Debany W.H., Hartmann C.R.P., Snethen T.* J. Algorithm for generating optimal tests for exclusive-OR nets // IEE Proc. – 1991. – Vol. 138. – No 2. – P. 93–96.
6. *Закревский А.Д., Закревский Л.А.* Диагностирование схем, построенных из элементов "сумма по модулю 2" // Докл. АН Беларуси. – 1996. – Т. 10. – № 6. – С. 7–11.
7. *Suprun V.P.* Fixed polarity Reed-Muller expressions of symmetric Boolean functions // Proc. IFIP WG 10.5 Workshop on Applications of the Reed-Muller Expansion in Circuit Design. – Makuhari, Chiba, Japan, – 1995. – P. 246–249.
8. *Кострикин А. И., Манин Ю. И.* Линейная алгебра и геометрия. – М.: Наука, 1986.
9. *Ланкастер П.* Теория матриц. – М.: Наука, 1978.
10. *Закревский А. Д.* К решению систем линейных логических уравнений с искаженными правыми частями – условия корректной постановки задачи // Информ. системы и технологии (IST'2002): Матер. I Междунар. конф., Минск, 5–8 ноября 2002 г. – Минск 2002. – Ч. 2. С. 45–50.
11. *Zakrevskij A.D.* Looking for shortest solutions of linear logical equations: theory and applications in logic design // 2. Workshop «Boolesche Probleme», 19/20 Sept., 1996. – Freiberg / Sachsen. P. 63–69.
12. *Закревский А.Д., Торопов Н.Р.* Полиномиальная реализация частичных булевых функций и систем. – Минск: Ин-т техн. кибернетики НАН Беларуси, 2001.
13. *Нильсон Н.* Искусственный интеллект. Методы поиска решений. – М.: Мир, 1971.
14. *Торопов Н.Р.* Ускорение лестничного алгоритма решения системы линейных логических уравнений // Методы логического проектирования / ОИПИ НАН Беларуси. – Минск, 2003. – Вып. 2 – С. 103–114.
15. *Gauss C.F.* Beitrage zur Theorie der algebraischen Gleichungen. – Gött, 1849.
16. *Закревский А.Д., Василькова И.В.* Быстрый алгоритм нахождения кратчайшего решения системы линейных логических уравнений // Методы логического проектирования / ОИПИ НАН Беларуси. – Минск, 2002. – Вып. 1. – С. 5–12.
17. *Закревский А.Д.* Декомпозиционные методы нахождения кратчайшего решения системы линейных логических уравнений // Докл. НАН Беларуси. – 2003. – Т. 47. – № 2. – С. 58–60.
18. *Закревский А.Д.* Оптимизация решений в линейном булевом пространстве – методы декомпозиции // Автоматика и вычислительная техника. – 2003. – № 5. – С. 28–36.
19. *Zakrevskij A.D., Zakrevski L.A.* Optimizing solutions in a linear Boolean space – a decomposition method // Proc. of STI '2003, Orlando, Florida, USA, July 2003. P. 276–280.
20. *Закревский А.Д., Василькова И.В.* Прогнозирование затрат времени на выполнение комбинаторных алгоритмов // Методы логического проектирования / ОИПИ НАН Беларуси. – Минск, 2003. – Вып. 2.

E-mail: zakr@newman.bas-net.by

☎ *(10-375-17) 288-65-99* ❑

ПРАВИЛА ОФОРМЛЕНИЯ СТАТЕЙ

ПРАВИЛА ОФОРМЛЕНИЯ СТАТЕЙ ДЛЯ ПУБЛИКАЦИИ В ЖУРНАЛЕ «ПРОБЛЕМЫ УПРАВЛЕНИЯ»

Статья представляется в редакцию на бумаге в 2-х экземплярах, с аннотацией и направлением организации, а также обязательно в электронном виде на дискете 3,5 дюйма или по электронной почте (не более 2 МБ). Аннотация, название статьи и фамилии авторов должны быть представлены также на английском языке. Примерный объем оригинальной статьи 12, обзорной – 18 страниц текста. Текст печатается через 2 интервала с одной стороны бумаги формата А4, страницы нумеруются. В электронной форме текст должен быть в редакторе не ниже Word 97 шрифтом № 12 Times New Roman; текст не форматируется, т.е. не имеет табуляции, колонок и т.д. Рисунки должны иметь расширение, совместимое с Word 97, или в формате CorelDraw: фотографии должны быть предельно четкими, черно-белыми, на глянцевой бумаге или в электронном виде TIFF с разрешением 300 dpi. Толщина линий рисунков, представляемых в электронной форме – не менее 3 пикселей.

Все буквенные обозначения, приведенные на рисунках, необходимо пояснять в основном или подрисуночном текстах (недопустимы повторные обозначения – в подрисуночных подписях и в тексте). Нумеровать следует только те формулы, на которые есть ссылка в последующем изложении. Список литературы (только органически связанной со статьей) составляется в порядке цитирования и дается в конце статьи. Ссылки на литературу в тексте отмечаются порядковыми номерами в квадратных скобках.

В конце статьи следует указать номер контактного телефона и электронный адрес. На отдельном листе дать сведения об авторах (полностью ф.и.о., ученая степень, должность и место работы, почтовый адрес и тел.)