ИНФОРМАТИКА

2004 октябрь-декабрь №4

УДК 518.5

А.Д. Закревский

РЕШЕНИЕ БОЛЬШИХ СИСТЕМ БУЛЕВЫХ УРАВНЕНИЙ

Многие проблемы анализа, синтеза и диагностики неисправностей логических схем, а также проблемы дедуктивного вывода, распознавания образов и защиты информации сводятся к решению комбинаторных задач с неизбежным перебором вариантов. Значительная часть таких задач формулируется на языке логических уравнений. В статье рассматриваются два класса систем, содержащих сотни булевых уравнений и переменных: большие системы логических уравнений с ограниченным числом переменных в каждом из них (БСЛУ) и системы линейных логических уравнений (СЛЛУ). Для решения таких систем в лаборатории логического проектирования ОИПИ НАН Беларуси разработана серия комбинаторных методов. Результаты испытаний этих методов на потоке псевдослучайных систем свидетельствуют об их высокой практической эффективности.

Введение

Большинство теоретических и практических проблем проектирования устройств логического управления [1], анализа и синтеза конечных автоматов и логических схем [2, 3], технической и медицинской диагностики [4], дедуктивного вывода типа доказательства теорем [5], защиты информации [6] и многих других сводится к решению комбинаторных задач [7–10]. Такие задачи формулируются в терминах различных разделов современной дискретной математики, например теории графов [11]. Не меньшее внимание уделяется приложениям логико-комбинаторных методов к проблеме распознавания образов [12–14].

Решение комбинаторных задач по необходимости связано с обширным перебором промежуточных решений. Как правило, это так называемые NP-трудные задачи, вычислительная сложность которых экспоненциально зависит от объема исходных данных [15]. Говорят, что для таких задач в принципе не существует эффективных алгоритмов решения. Однако при этом подразумевается *теоретическая* эффективность: предполагается, что алгоритм эффективен, если и только если его вычислительная сложность ограничена некоторым полиномом от объема исходных данных при неограниченном росте этого объема. Между тем можно разрабатывать *практически* эффективные алгоритмы решения многих NP-трудных задач, с успехом используемые при реальных ограничениях на объем исходных данных и позволяющие получать приближенные решения, достаточно близкие к оптимальным [16].

Адекватным математическим аппаратом для многих комбинаторных задач служит формализм логических уравнений [17–22]. Логическим уравнением называется выражение вида $A = B$, где A и B – некоторые формулы алгебры логики, составленные из символов логических переменных и операций, а также скобок, определяющих порядок действия последних. Аппарат логических уравнений оказался весьма универсальным, будучи успешно применен для решения задач автоматизированного проектирования и управления [23], синтеза и анализа многоуровневых комбинационных схем (в частности, программируемых логических матриц [24–26]), распознавания образов [27], полисиллогистики [28, 29] и теории логического вывода [30].

Решить уравнение – значит найти его корни (комбинации значений переменных, на которых A и B принимают равные значения). Иногда требуется найти один (любой) корень, иногда все (либо убедиться в их отсутствии). Представляет интерес нахождение оптимального в некотором смысле корня. Различные варианты этой задачи и методы ее решения (см. [22]) рассматривались в лаборатории логического проектирования ОИПИ НАН Беларуси, где были продолжены изыскания, начатые в Томском государственном университете и Сибирском физико-техническом институте в начале 1960-х гг. [31–33].

В последние годы основное внимание уделялось разработке практически эффективных методов решения больших систем логических уравнений, составленных из сотен и тысяч уравнений и переменных. Ниже кратко описываются результаты этих исследований.

1. Большие системы логических уравнений. Минимизация дерева поиска

Особый интерес представляют системы логических уравнений ($A_1 = 1$, $A_2 = 1$, …, $A_m = 1$), эквивалентные одному конъюнктивному уравнению $A_1 \wedge A_2 \wedge \ldots \wedge A_m = 1$. Как правило, в системах логических уравнений, которые приходится решать при анализе сложных логических структур, фигурирует много переменных (десятки, сотни и даже тысячи). Однако их число в каждом из уравнений может быть сильно ограничено, например не превышать десяти. Будем называть такие системы *большими* (БСЛУ).

Любую БСЛУ с булевыми переменными, образующими общее упорядоченное множество $\boldsymbol{x} = (x_1, x_2, ..., x_n)$, нетрудно преобразовать к виду $F = (\varphi_1(\boldsymbol{u}_1) = 1, \varphi_2(\boldsymbol{u}_2) = 1, ..., \varphi_m(\boldsymbol{u}_m) = 1)$, где $\varphi_i(\boldsymbol{u}_i)$ – некоторые булевы функции с аргументами, выбираемыми из множества $\boldsymbol{x}$: $\boldsymbol{u}_i \subseteq \boldsymbol{x}$, $i = 1, 2, ..., m$. Решение системы F заключается в поиске ее корней – наборов значений переменных $x_1, x_2, ..., x_n$, обращающих в 1 каждую функцию φ_i.

Автоматизируя процесс решения, удобно представлять каждую функцию $\varphi_i(\boldsymbol{u}_i)$ парой булевых векторов: 2^k-компонентным *вектором значений* $\boldsymbol{v}_i$, перечисляющим значения функции на всех наборах значений ее k аргументов, и n-компонентным *вектором аргументов* $\boldsymbol{w}_i$, в котором единицами отмечены переменные, играющие роль аргументов функции $\varphi_i(\boldsymbol{u}_i)$ (и, следовательно, образующие множество $\boldsymbol{u}_i$). Систему уравнений в целом зададим *матрицей функций* $\boldsymbol{V}$ и *матрицей аргументов* $\boldsymbol{W}$, их строки характеризуют отдельные уравнения.

Например, пара матриц

$$\boldsymbol{V} = \begin{matrix} 1\,0\,0\,1 & 1\,1\,0\,1 & 1\,1\,1\,0 & 0\,0\,0\,0 \\ 1\,1\,1\,0 & 0\,0\,0\,0 & 1\,1\,0\,1 & 0\,1\,1\,1 \\ 0\,0\,0\,0 & 1\,1\,1\,0 & 1\,1\,1\,0 & 0\,1\,0\,1 \\ 0\,0\,0\,0 & 0\,1\,1\,1 & 0\,0\,1\,1 & 1\,1\,1\,0 \\ 0\,1\,0\,0 & 1\,0\,1\,0 & 0\,0\,1\,0 & 1\,1\,0\,1 \end{matrix}, \qquad \begin{matrix} a\ b\ c\ d\ e\ f\ g\ h \\ \boldsymbol{W} = \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} 1\,1\,1\,0\,0\,0\,1\,0 \\ 0\,0\,0\,0\,1\,1\,1\,1 \\ 0\,0\,1\,1\,1\,0\,0\,1 \\ 1\,1\,1\,1\,0\,0\,0\,0 \\ 1\,1\,0\,0\,1\,1\,0\,0 \end{matrix}$$

представляет систему логических уравнений с пятью функциями $\varphi_1(a, b, c, g)$, $\varphi_2(e, f, g, h)$, $\varphi_3(c, d, e, h)$, $\varphi_4(a, b, c, d)$ и $\varphi_5(a, b, e, f)$, и первая из этих функций принимает значение 1 на следующих восьми наборах значений ее аргументов a, b, c, g: 0000, 0011, 0100, 0101, 0111, 1000, 1001, 1010.

Практический интерес представляет рассмотрение систем с малым числом корней. К этому случаю относится задача проверки системы на выполнимость, т. е. на наличие хотя бы одного корня (к ее решению сводятся многие задачи логического проектирования дискретных систем), а также задача нахождения единственного корня системы (типичная для диагностики и распознавания). Прямые методы решения этих задач (перебор и испытание всех наборов или логическое перемножение всех функций φ_i) с ростом m и n становятся практически нереализуемыми. Более перспективным является подход, основанный на обходе дерева поиска, строящегося в процессе решения [34].

В работах [35–38] были предложены два способа реализации этого подхода, основанные на эффективном использовании информации, содержащейся в матрице аргументов $\boldsymbol{W}$.

Минимизация дерева поиска в методе перебора уравнений. Рассматриваемый метод восходит к методу Черри и Васвани [39], который сводится к перемножению совершенных ДНФ характеристических функций $\varphi_i(\boldsymbol{u}_i)$, представляемых строками матрицы $\boldsymbol{V}$. Этот метод был улучшен предварительным упорядочением уравнений, а также преобразованием совершенных ДНФ в эквивалентные ортогональные ДНФ, по возможности минимизированные.

В данном методе рассматривается дерево поиска T_e, ярусы которого ставятся во взаимно однозначное соответствие с уравнениями системы. На i-м ярусе расположены вершины, представляющие корни подсистемы, составленной из первых i уравнений. Этими корнями служат некоторые элементы булева пространства переменных множества $\boldsymbol{u}_1 \cup \boldsymbol{u}_2 \cup \ldots \cup \boldsymbol{u}_i$. Обозначим через $r(i)$ мощность этого множества.

Утверждение 1. Математическое ожидание $M_e(i)$ числа вершин дерева T_e на ярусе i равно $p^i 2^{r(i)}$.

В частности, число вершин последнего яруса, представляющих корни системы в целом, оценивается формулой $M_e(m) = p^m 2^n$.

При включении в подсистему следующего по порядку уравнения, представленного функцией $\varphi_{i+1}(\boldsymbol{u}_{i+1})$, множество рассматриваемых аргументов может расширяться за счет таких переменных этой функции, которые не встречались в предшествующих функциях. В результате число корней возрастает. С другой стороны, множество корней системы равно пересечению множеств корней составляющих ее уравнений, что может приводить к уменьшению числа корней с ростом числа уравнений. Суммарный эффект выражается следующей формулой.

Утверждение 2. $M_e(i+1) = M_e(i)\, p\, 2^{r(i+1)-r(i)}$.

Как видно из данной формулы, рост числа вершин при переходе на следующий ярус зависит не от общего числа аргументов функции $\varphi_{i+1}(\boldsymbol{u}_{i+1})$, а лишь от числа тех из них, которые не встречались ранее.

Общий объем вычислений при поиске всех корней системы пропорционален величине M_e – суммарному числу вершин дерева T_e : $M_e = M_e(1) + M_e(2) + \ldots + M_e(m)$. Число вершин последнего яруса определяется однозначно, однако числа вершин на остальных ярусах, а также общее число вершин M_e существенно зависят от выбранного порядка рассмотрения уравнений.

Предлагается следующее правило сокращения объема вычислений, вытекающее из утверждения 2: *очередное уравнение должно содержать минимальное число новых переменных*.

Утверждение 3. Существует пример (значения 0 и 1 в матрице $\boldsymbol{V}$ равновероятны; $m = n$; $w_i^j = 1$, если $i \le j$, и $w_i^j = 0$ в противном случае), для которого при исходном порядке уравнений дерево поиска содержит 2^n вершин, а при оптимальном – только n.

Минимизация дерева поиска в методе перебора аргументов. В этом методе строится дерево поиска T_a, которое показывает иерархию бифуркаций по значениям булевых аргументов, образующих последовательность $x_1, x_2, \ldots, x_n$ и поставленных во взаимно однозначное соответствие с n ярусами дерева. На j-м ярусе располагаются вершины, представляющие все такие наборы значений переменных $x_1, x_2, \ldots, x_i$, которые не обращают в 0 ни одну из функций системы. Обозначим через $M_a(j)$ математическое ожидание числа таких вершин.

Утверждение 4. $M_a(j) = 2^j \prod_{i=1}^{m} S(p, q(i,j))$, где $S(p, r) = 1 - (1-p)^{2^r}$ – вероятность того, что случайная функция с r аргументами и параметром p не обратится в 0, а $q(i,j)$ – число единиц в i-й строке матрицы $\boldsymbol{W}$, расположенных правее компоненты j .

Число вершин на последнем ярусе, равное числу корней системы, прогнозируется формулой $M_a(n) = 2^n p^m$, а число всех вершин дерева T_a определяется формулой $M_a = M_a(1) + M_a(2) + \ldots + M_a(n)$. С ростом числа r аргументов случайной булевой функции вероятность $S(p, r)$ непротиворечивости рассматриваемого уравнения быстро стремится к единице. Например, при $p = 0{,}5$ она вычисляется по формуле $1 - 2^{-2^r}$, получая следующие значения: $S(0) = 1/2$, $S(1) = 3/4$, $S(2) = 15/16$, $S(3) = 255/256$, $S(4) = 65535/65536$ и т. д. Очевидно, что при практических расчетах можно положить $S(r) = 1$, если $r > 3$.

Предлагаемый метод заключается в оптимизации порядка выбора аргументов, ставящихся в соответствие ярусам дерева. При этом минимизируется математическое ожидание числа вершин в последовательно рассматриваемых ярусах дерева.

Утверждение 5. При переходе от яруса $j - 1$ к ярусу j математическое ожидание числа вершин в ярусе увеличивается в $2 / \prod R(q(i,j))$ раз, где произведение берется по всем i, для которых $w_i^j = 1$.

Оба метода были запрограммированы в C++ и испытаны на PC COMPAC Presario (процессор Intel Pentium III, 1000 МГц), на псевдослучайных БСЛУ. Эксперименты показали, что метод перебора уравнений не в состоянии конкурировать с методом перебора аргументов. Например, на решение системы с параметрами $n = 40$, $m = 38$, $k = 5$ первый метод затратил 633 с, а второй – всего 3 с.

2. Редукция БСЛУ

Эффективность комбинаторного поиска по дереву можно существенно повысить, сокращая область поиска. Были разработаны три способа такого сокращения, названные *методом распространения констант*, *методом локальной редукции* и *методом силлогизмов*.

Суть предлагаемых методов заключается в последовательном анализе функций $\varphi_i(\boldsymbol{u}_i)$ и поиске так называемых k-запретов, или запретов ранга k – некоторых наборов значений k переменных, на которых функция обращается в 0. Найденные запреты подставляются в другие уравнения системы F, в результате чего отдельные функции $\varphi_i(\boldsymbol{u}_i)$ преобразуются путем редукции их характеристических множеств M_i: они меняют свои значения на некоторых наборах аргументов с 1 на 0. При этом сохраняется множество корней системы F в целом [40, 41].

Метод распространения констант. Этот метод описан в работе [42] и представляет собой развитие метода «отраженных волн», предложенного в [43]. Он применим в случае, когда значения некоторых переменных в рассматриваемой системе определяются однозначно уже в пределах одного уравнения, так как находятся запреты первого ранга.

Идея данного метода довольно проста и отображается следующими правилами поиска и распространения констант в системе F, легко реализуемыми при векторном представлении рассматриваемых функций и редуцирующими систему с сохранением множества ее корней**.**

Пусть φ_i и φ_j – некоторые функции из системы F, зависящие от аргумента x_p.

Утверждение 6. Если $x_p\varphi_i = 0$, то в систему можно добавить уравнение $x_p = 0$, фиксирующее значение переменной x_p; если $x_p'\varphi_i = 0$, то аналогично фиксируется значение $x_p = 1$.

Утверждение 7. При наличии в системе уравнения $x_p = 0$ функцию φ_j можно заменить на $x_p\varphi_j'$ (на $x_p\varphi_j$ при $x_p = 1$).

Утверждение 8. При наличии в системе пары уравнений $x_p = 1$ и $x_p = 0$ обнаруживается противоречивость системы (она не имеет корней).

Утверждение 6 можно использовать для выявления констант в системе, а утверждение 7 – для их распространения. Процесс распространения констант имеет цепной характер и при определенных условиях идет до конца – находится единственный корень рассматриваемой системы логических уравнений.

Метод распространения констант удалось применить к решению одной задачи криптоанализа. В работе [44] рассматривалась роторная шифровальная электромеханическая машина Hagelin M-209, которая использовалась немцами во время второй мировой войны. Было показано, что ее криптоанализ сводится к решению определенной системы из многих логических уравнений (порядка 500), каждое из которых содержит шесть булевых переменных, между тем как их общее число равно 131 – набор значений этих переменных образует искомый ключ. Применение метода распространения констант к решению этой задачи оказалось весьма эффективным. В сравнении с предложенным в [44] методом, основанным на использовании бинарных таблиц решений, время нахождения ключа сократилось более чем в тысячу раз.

Метод локальной редукции. Данный метод, предложенный в работе [45], не ограничивается поиском и распространением констант, т. е. 1-запретов, а рассматривает также по мере возможности запреты более высокого ранга. Локальность метода выражается в том, что возможность редуцирования выявляется при рассмотрении всевозможных пар функций $\varphi_i(\boldsymbol{u}_i)$ и $\varphi_j(\boldsymbol{u}_j)$ с пересекающимися множествами аргументов: $\boldsymbol{u}_{i,j} = \boldsymbol{u}_i \cap \boldsymbol{u}_j \neq \varnothing$.

Введем ряд обозначений. Рассмотрим характеристическое множество M_i функции $\varphi_i(\boldsymbol{u}_i)$ в пространстве аргументов, составляющих множество $\boldsymbol{u}_i$, и некоторый его элемент $\boldsymbol{a}$: $\boldsymbol{a} \in M_i$. Этот элемент представляет собой k-компонентный булев вектор, где k – число переменных в множестве аргументов функции $\varphi_i(\boldsymbol{u}_i)$: $k = |\boldsymbol{u}_i|$. Далее, пусть $\boldsymbol{v}$ – некоторое подмножество из $\boldsymbol{u}_i$: $\boldsymbol{v} \subseteq \boldsymbol{u}_i$. Обозначим через $\boldsymbol{a}/\boldsymbol{v}$ *проекцию элемента* $\boldsymbol{a}$ *на* $\boldsymbol{v}$, т. е. совокупность значений переменных из множества $\boldsymbol{v}$, представленных соответствующими компонентами вектора $\boldsymbol{a}$. Множество всех различных проекций элементов из M_i на $\boldsymbol{v}$ назовем *проекцией множества* M_i *на* $\boldsymbol{v}$ и обозначим $M_i/\boldsymbol{v}$. Через $M_{i,j}$ обозначим пересечение множеств $M_i/\boldsymbol{u}_{i,j}$ и $M_j/\boldsymbol{u}_{i,j}$, а через $M_{i/j}$ – множество всех таких элементов из M_i, проекции которых на $\boldsymbol{u}_{i,j}$ принадлежат множеству $M_{i,j}$.

Пусть, например, $\boldsymbol{u}_i = (a, b, c, d, e)$, $\boldsymbol{u}_j = (c, d, e, f, g, h)$, $M_i = (01101, 11010, 10011)$, $M_j = (101110, 001101, 010010)$. Тогда $\boldsymbol{u}_{i,j} = \boldsymbol{u}_{j,i} = (c, d, e)$, $M_{i,j} = M_{j,i} = (101, 010)$, $M_{i/j} = (01101, 11010)$, $M_{j/i} = (101110, 010010)$.

Введем в рассмотрение операцию $M_i := M_{i/j}$ замены M_i на $M_{i/j}$.

Утверждение 9. При любых $i, j = 1, 2, ..., m$ операция $M_i := M_{i/j}$ является эквивалентным преобразованием системы F, сохраняющим множество ее корней.

Будем говорить, что операция $M_i := M_{i/j}$ *применима* к упорядоченной паре функций (φ_i, φ_j), если $M_i \neq M_{i/j}$. Вероятность применимости растет с ростом мощности множества $\boldsymbol{u}_{i,j}$ и уменьшается с ростом мощности множества M_j, будучи достаточно высокой, например, при выполнении отношения $|M_j| < 2^s$, где $s = |\boldsymbol{u}_{i,j}|$.

Рассмотрим процесс последовательного выполнения этой операции на парах, к которым она применима. Он может завершиться редуцированием некоторого из множеств M_i до пустого множества, что будет свидетельствовать о невыполнимости системы F, или получением тупикового набора функций, в котором данная операция не применима ни к одной паре. Назовем этот процесс *локальным редуцированием системы* F.

Особый интерес представляют такие случаи применимости операции $M_i := M_{i/j}$, как ситуации сильного пересечения множеств $\boldsymbol{u}_i$ и $\boldsymbol{u}_j$ (особенно когда $\boldsymbol{u}_i = \boldsymbol{u}_j$) и наличия функций φ_j с малым числом единичных значений. Подобные ситуации типичны для задач распознавания и криптографии.

Программная реализация и испытания на большом потоке псевдослучайных примеров показали высокую эффективность метода локальной редукции и его применимость к системам, содержащим тысячу и более уравнений и переменных [46]. В то же время был экспериментально обнаружен *эффект взрыва*, наблюдаемый при определенном отношении между параметрами системы. Он приводит к *коллапсу* системы – резкому сокращению числа корней во всех уравнениях.

Метод силлогизмов. Этот метод описан в работе [47]. Подобно предыдущим, он позволяет решать большие системы логических уравнений с ограниченным числом переменных в каждом из них. Метод заключается в анализе по циклу отдельных уравнений, выявлении запретов второго ранга (эквивалентных категорическим суждениям силлогистики Аристотеля), выводе всех логических следствий и их использовании для сокращения числа корней в других уравнениях.

Метод комбинирует технику логического вывода полисиллогизмов [29] с методом распространения констант. Его идея заключается в последовательном анализе уравнений и выявлении системы импликативных связей между отдельными аргументами, запрещающих некоторые наборы значений на парах переменных, а также нахождении всех логических следствий данной системы.

Допустим, что при последовательном анализе уравнений, образующих систему F, выявлено некоторое множество 2-запретов P. Рассмотрим задачу его замыкания, т. е. добавления в это множество всех других 2-запретов, логически следующих из P. Эта задача эквивалентна задаче полисиллогистики: для заданного множества суждений-посылок найти все суждения-следствия. Полученное замкнутое множество запретов обозначим через $Cl(P)$. В работе [47] предложен метод получения этого множества, отличающийся от известного метода резолюций и его графического варианта [29] применением векторно-матричных операций, позволяющих параллелить операции логического вывода.

Обозначим через X_t^1 и X_t^0 множества всех литералов, входящих в перечисленные в P запреты совместно с литералом x_t или x_t' соответственно. Введем в рассмотрение оператор Cl_t *частичного замыкания множества* P *по переменной* x_t, расширяющий это множество путем его объединения с прямым произведением $X_t^1 \times X_t^0$, содержащим результаты всевозможных резолюций по этой переменной.

Утверждение 10. $Cl_t(P) = P \cup X_t^1 \times X_t^0 \subseteq Cl(P)$.

Утверждение 11. $Cl(P) = Cl_1 Cl_2 \ldots Cl_n (P)$.

Из этих утверждений следует способ замыкания множества запретов P путем его последовательного частичного замыкания по отдельным переменным.

3. Системы линейных логических уравнений. Решение неопределенных СЛЛУ

Исследования линейных логических уравнений и систем были начаты в лаборатории логического проектирования в связи с разработкой методов синтеза логических схем в базисе AND/EXOR, включающем элемент суммирования по модулю 2 [48].

Будем рассматривать неоднородную систему из m линейных логических уравнений с n переменными, называя ее СЛЛУ:

$$a_1^{\,1}x_1 \oplus a_1^{\,2}x_2 \oplus \ldots \oplus a_1^{\,n}x_n = y_1\ ;$$
$$a_2^{\,1}x_1 \oplus a_2^{\,2}x_2 \oplus \ldots \oplus a_2^{\,n}x_n = y_2\ ;$$
$$\ldots$$
$$a_m^{\,1}x_1 \oplus a_m^{\,2}x_2 \oplus \ldots \oplus a_m^{\,n}x_n = y_m\ .$$

Система может оказаться *неопределенной* (обладая двумя или более корнями), *определенной* (с единственным корнем) или *переопределенной* (не имея корней). Неопределенные и определенные системы называются также *совместными*, переопределенные – *несовместными*.

Искать корни можно известным методом исключения переменных, предложенным Гауссом для линейных алгебраических уравнений с действительными переменными и модифицируемым применительно к логическим уравнениям. Данный метод определяет, совместна ли система, находит единственный корень определенной системы либо множество всех корней неопределенной системы. Полученный при этом результат является исчерпывающим в случае определенной системы, однако он недостаточен, когда система оказывается неопределенной либо переопределенной.

При рассмотрении ряда практических проблем возникают задачи нахождения оптимального решения СЛЛУ. Например, при синтезе логических AND/EXOR-схем приходится искать кратчайший корень неопределенной системы, представляемый набором с минимальным числом единиц. Для решения этой задачи был разработан практически эффективный алгоритм, названный *лестничным*, а также алгоритм, опирающийся на метод Гаусса [49, 50]. В случае несовместной системы можно поставить задачу удовлетворения максимального числа уравнений, которая представляет интерес с точки зрения теории и практики передачи и защиты информации. Ниже устанавливается связь между этими двумя задачами и предлагаются эффективные комбинаторные методы их решения.

Канонизация неопределенной системы методом Гаусса и нахождение кратчайшего корня. Представим систему в виде матрично-векторного уравнения $A\,x = y$, где $\boldsymbol{A}$ – булева $(m\times n)$-матрица коэффициентов, $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ – булев вектор неизвестных, $\boldsymbol{y} = (y_1, y_2, \ldots, y_m)$ – булев вектор свободных членов. Покомпонентная сумма (по модулю два) столбцов матрицы, соответствующих переменным, принимающим в решении значение 1, должна быть равна заданному вектору $\boldsymbol{y}$. Добавив справа к матрице коэффициентов $\boldsymbol{A}$ (принято называть ее основной) вектор-столбец $\boldsymbol{y}$, получим их конкатенацию – расширенную матрицу $\boldsymbol{S} = (\boldsymbol{A}, \boldsymbol{y})$. Она представляет известную о системе информацию.

Суть метода Гаусса заключается в канонизации матрицы $\boldsymbol{S}$. Допустим, что $n > m$ и строки матрицы $\boldsymbol{S}$ линейно независимы. Тогда в матрице $\boldsymbol{A}$ выбирается некоторая максимальная совокупность m линейно независимых столбцов и преобразуется в *базис* – квадратный столбцовый минор с одной единицей в каждом ряду. Остальные $n - m$ столбцов матрицы $\boldsymbol{A}$ преобразуются в *остаток*. Каждое из 2^{n-m} подмножеств столбцов из остатка определяет некоторое решение рассматриваемой системы, в которое кроме данного подмножества включаются некоторые столбцы базиса, что обеспечивает равенство суммы всех включенных столбцов столбцу $\boldsymbol{y}$ (также в преобразованном виде). Число столбцов, вошедших в решение, назовем его *весом*.

При поиске кратчайшего решения по методу, описанному в работах [49, 50], производится перебор подмножеств столбцов в остатке, рассмотрение определяемых ими решений и выбор кратчайшего из них. Если известно, что вес кратчайшего решения не превышает некоторого значения γ, то уровень перебора (мощность перебираемых подмножеств) ограничивается этой величиной. Если $\gamma \ge n - m$, приходится перебирать все 2^{n-m} подмножеств.

Ожидаемый вес кратчайшего решения системы уравнений с параметрами m и n можно оценить еще до нахождения самого решения, представив его как функцию $\gamma(m, n)$. Начнем с

вычисления математического ожидания $\alpha(m, n, k)$ числа решений с весом k. Положим, что рассматриваемая система случайна – каждый элемент булевой матрицы $\boldsymbol{A}$ принимает значение 1 с вероятностью 0,5. Тогда вероятность того, что случайно выбранное подмножество столбцов в матрице $\boldsymbol{A}$ окажется решением, равна 2^{-m} (вероятность равенства двух случайных булевых векторов длины m).

Утверждение 12. $\alpha(m, n, k) = C_n^k 2^{-m}$, где C_n^k – число сочетаний из n по k.

Аналогично через $\beta(m, n, k)$ обозначим математическое ожидание числа решений, вес которых не превышает k.

Утверждение 13. $\beta(m, n, k) = \sum_{i=0}^{k} C_n^i 2^{-m}$.

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

Утверждение 14. $\gamma(m, n) = k$, где $\beta(m, n, k) < 1 \le \beta(m, n, k+1)$.

Оценивая объем вычислений при поиске кратчайшего решения, положим, что он пропорционален числу N рассматриваемых подмножеств столбцов остатка. Очевидно, значение N в сильной степени зависит от уровня перебора, равного $\gamma(m, n)$.

Утверждение 15. $N(m, n) = \sum_{i=0}^{\gamma(m, n)} C_{n-m}^i$.

Следствие. Если $\gamma(m, n) \ge n - m$, то $N(m, n) = 2^{n-m}$.

Метод непересекающихся остатков. Заметим, что экспоненциальный рост числа $N(m, n)$ с ростом разности $n - m$ существенно ограничивает практическую применимость описанного выше метода. Это обстоятельство послужило причиной разработки нового метода.

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 $n \ge q(n - m)$. Построим множество Q, составленное из q соответствующих канонических форм, базисы которых строятся на основе указанных совокупностей, и будем искать оптимальное решение в рамках этих форм, последовательно увеличивая уровень перебора до некоторой величины.

Утверждение 16. В множестве Q найдется каноническая форма, где кратчайшее решение можно найти на уровне перебора не выше $\lfloor \gamma/q \rfloor$ – целого, ближнего снизу к частному γ/q.

Утверждение 17. Кратчайшее решение системы можно найти путем последовательного рассмотрения остатков канонических форм из множества Q, ограничивая уровень перебора подмножеств в этих остатках величиной $\lfloor (w\text{-}1)/q \rfloor$, где w – вес кратчайшего из решений, найденных к текущему моменту времени.

Эти утверждения образуют основу для предложенного в работах [51, 52] декомпозиционного метода нахождения кратчайшего решения системы линейных логических уравнений. Обратим внимание на существенное понижение уровня перебора подмножеств в этом методе. В нем реализуется перебор подмножеств во всех остатках сначала на уровне 0, затем на уровне 1 и т. д., пока этот уровень не превысит $\lfloor (w\text{-}1)/q \rfloor$.

Утверждение 18. Число $N_r(m, n)$ подмножеств столбцов, рассматриваемых в методе непересекающихся остатков, определяется следующей формулой:

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

Предложенный метод позволяет существенно сократить время поиска кратчайшего решения в сравнении с методом Гаусса. Например, при $n = 100$ и $m = 70$ вес кратчайшего решения оказывается равным 20. В методе, основанном на рассмотрении одной канонической формы, в соответствии с утверждением 15 перебирается около 1 050 млн подмножеств из остатка. В ме-

тоде непересекающихся остатков (см. утверждение 18) рассматривается только около 2,3 млн подмножеств в трех остатках, т. е. в 456 раз меньше.

Методы распознавания и рандомизации. В практических ситуациях может существовать *короткое* решение с весом, меньше веса γ кратчайшего корня для заведомо случайной системы. Тогда решение СЛЛУ можно существенно ускорить, применив метод распознавания, при котором поиск оптимального решения прекращается, как только вес w очередного решения окажется меньше некоторого заданного порога (например, если $w < \gamma - 1$) [53, 54].

Заметим, что столбцы матрицы $\boldsymbol{A}$, составляющие кратчайшее решение, распределяются по остаткам канонических форм случайно, т. е. не равномерно, а с некоторой дисперсией. Благодаря этому найдется остаток с минимальным числом k вошедших туда столбцов, заметно меньшим, чем $\lfloor \gamma/q \rfloor$. Это позволяет, объединив методы декомпозиции и распознавания, понизить уровень перебора k и тем самым значительно ускорить поиск решения.

Еще большего ускорения можно добиться, применив метод рандомизации [55]. На базе случайного выбора из матрицы $\boldsymbol{A}$ различных максимальных множеств линейно независимых столбцов строится некоторое множество канонических форм. С большой вероятностью расход времени на их построение и анализ с лихвой компенсируется снижением значения параметра k, задающего уровень производимого перебора.

В результате получается выигрыш, хорошо иллюстрируемый результатами компьютерного эксперимента (C++, PC COMPAC Presario – процессор Intel Pentium III, 1000 МГц). Решалась серия из 30 случайных СЛЛУ с параметрами $m = 625$, $n = 700$, обладающих коротким решением ($w = 105$). Среднее время решения примера методом непересекающихся остатков в комбинации с методом распознавания составило 97 дней. Аналогичная величина для обобщенного декомпозиционного метода, использующего 250 различных канонических форм, оказалась равной 11 ч. Таким образом, поиск решений ускорился примерно в 200 раз. Заметим, что для оценки времени решения использовался метод прогноза, предложенный в работе [56].

4. Решение несовместных СЛЛИ

Как уже было сказано, несовместными называются СЛЛУ $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{y}$, не имеющие корней (обычно в таких системах $m > n$). Тем не менее, можно решать и такие уравнения.

Постановки задач над несовместными СЛЛУ. Положим, что $m > n$, а величины $\boldsymbol{A}$ и $\boldsymbol{y}$ случайны, причем все столбцы матрицы $\boldsymbol{A}$ линейно независимы. Тогда среди 2^m значений булева m-вектора $\boldsymbol{y}$ найдутся 2^n значений (соответствующих различным значениям n-вектора $\boldsymbol{x}$), подстановка которых вместо $\boldsymbol{y}$ делает систему совместной. Такие значения называются ниже *пригодными*.

Сформулируем три задачи над несовместной системой $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{y}$ с заданными $\boldsymbol{A}$ и $\boldsymbol{y}$, комбинаторно эквивалентные друг другу. Все они одинаково сложны с вычислительной точки зрения, и решение любой из них нетрудно трансформировать в решение другой. Можно считать, что они представляют различные интерпретации одной и той же комбинаторной задачи.

Нахождение ближайшего к вектору $\boldsymbol{y}$ пригодного значения. Изменить в векторе $\boldsymbol{y}$ значения минимального числа компонент, с тем чтобы полученное значение вектора оказалось пригодным.

Решение максимального числа уравнений. Найти значение вектора $\boldsymbol{x}$, удовлетворяющее максимальному числу уравнений в системе (обращающее их в тождества).

Линейная аппроксимация частичной булевой функции. Рассматривая систему из матрицы $\boldsymbol{A}$ и вектора $\boldsymbol{y}$ как частичную булеву функцию $y(\boldsymbol{x})$, определенную на строках матрицы $\boldsymbol{A}$, найти ее оптимальную аппроксимацию линейной функцией, совпадающей по значениям с функцией $y(\boldsymbol{x})$ на максимальном числе строк.

В работах [57, 58] показано, что решение этих задач для несовместной системы с m уравнениями и n переменными сводится к нахождению кратчайшего корня неопределенной системы с $n - m$ уравнениями и m переменными, получаемой из исходной системы. Ниже излагаются основные положения соответствующего метода.

Метод отображения в нуль-пространство. Будем говорить, что совокупность булевых векторов линейно связана, если их покомпонентная сумма по модулю два равна нулевому вектору $\boldsymbol{0}$.

Утверждение 19. Каждой линейной связанной совокупности строк матрицы $\boldsymbol{A}$ совместной системы линейных логических уравнений $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{y}$ должна соответствовать линейно связанная совокупность компонент вектора $\boldsymbol{y}$.

Зададим линейно связанные совокупности строк матрицы $\boldsymbol{A}$ булевыми m-векторами, соразмерными с вектором $\boldsymbol{y}$ (отмечая единицами элементы совокупности). Множество таких векторов обозначим через L.

Утверждение 20. Множество L представляет собой множество всех решений матричного уравнения $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{x} = \boldsymbol{0}$, где $\boldsymbol{A}^{\mathrm{T}}$ – результат транспонирования матрицы $\boldsymbol{A}$.

Утверждение 21. Множество L является нуль-пространством над множеством столбцов матрицы $\boldsymbol{A}$.

Напомним, что нуль-пространством над некоторым множеством булевых векторов называется множество всех векторов, ортогональных каждому вектору из данного множества. Два вектора считаются ортогональными, если их покомпонентная конъюнкция содержит четное число единиц.

Обозначим через L_{bas} некоторый базис множества L, представив его матрицей $\boldsymbol{L}_{bas}$, составленной из $m - n$ строк. Каждое из $2^{m-n} - 1$ непустых подмножеств этого базиса порождает (посредством операции суммирования) соответствующую линейно связанную совокупность.

Утверждение 22. Если условие из утверждения 19 выполняется для всех элементов базиса, то оно выполняется и для всех линейно связанных совокупностей.

Утверждение 23. Базисную матрицу $\boldsymbol{L}_{bas}$ можно получить по формуле $\boldsymbol{L}_{bas} = \mathrm{Conv}(\mathrm{Can}(\boldsymbol{A}^{\mathrm{T}}))$, где через $\boldsymbol{A}^{\mathrm{T}}$ обозначен результат транспонирования булевой $(m \times n)$-матрицы $\boldsymbol{A}$, Can – оператор канонизации матрицы по методу Гаусса, а Conv – оператор конверсии полученной канонической формы.

Оператор Can получает каноническую форму с $m - n$ столбцами в остатке, а оператор Conv строит булеву матрицу с таким же числом строк, которые находятся при анализе соответствующих столбцов остатка и представляют некоторые решения матричного уравнения $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{x} = \boldsymbol{0}$. Каждая из этих строк содержит единицу в компоненте, соответствующей выбранному столбцу остатка, и нули во всех компонентах, соответствующих другим столбцам остатка. Значения остальных компонент определяются набором единиц в соответствующем столбце остатка: выбираются строки канонической формы, содержащие данные единицы, а затем столбцы базиса с единицами в этих строках. Компоненты конструируемой строки, соответствующие выбранным таким образом столбцам базиса, получают значение 1.

В случае, когда каноническая форма нормализована (базис представлен единичной матрицей $\boldsymbol{I}$ с единицами на главной диагонали), конверсия выполняется особенно легко.

Утверждение 24. Если каноническая форма задана в виде блочной матрицы $[\boldsymbol{I}_n, \boldsymbol{R}]$, где $\boldsymbol{I}_n$ – единичная $(n \times n)$-матрица базиса, а $\boldsymbol{R}$ – $(n \times (m-n))$-матрица остатка, то результат конверсии будет представлен блочной матрицей $[\boldsymbol{R}^{\mathrm{T}}, \boldsymbol{I}_{m-n}]$, где $\boldsymbol{R}^{\mathrm{T}}$ – результат транспонирования матрицы $\boldsymbol{R}$, а $\boldsymbol{I}_{m-n}$ – единичная $((m-n)\times(m-n))$-матрица.

Введем в рассмотрение булев вектор-синдром $\boldsymbol{s}$, содержащий информацию о том, в какой степени вектор $\boldsymbol{y}$ не удовлетворяет требованиям утверждения 19. Он составлен из $m - n$ компонент, соответствующих строкам матрицы $\boldsymbol{L}_{bas}$. Их единичные значения означают, что для соответствующих линейно связанных совокупностей условие утверждения 19 не выполняется.

Утверждение 25. $\boldsymbol{s} = \boldsymbol{L}_{bas}\,\boldsymbol{y}$.

Задача заключается в том, чтобы путем изменения значений минимального числа компонент вектора $\boldsymbol{y}$ получить пригодный вектор $\boldsymbol{y}^*$, удовлетворяющий требованиям утверждения 19 для всех строк матрицы $\boldsymbol{L}_{bas}$, т. е. обращающий в нуль вектор-синдром $\boldsymbol{s}$. Другими словами, надо найти такой вектор $\boldsymbol{u}$ с минимальным числом единичных компонент, чтобы выполнялось отношение $\boldsymbol{L}_{bas}\,(\boldsymbol{y} \oplus \boldsymbol{u}) = \boldsymbol{0}$, а значит, и отношение $\boldsymbol{L}_{bas}\,\boldsymbol{u} = \boldsymbol{s}$. Отсюда следует

Утверждение 26. Искомый вектор $\boldsymbol{u}$ представляет собой кратчайшее решение неопределенной СЛЛУ $\boldsymbol{L}_{bas}\,\boldsymbol{x} = \boldsymbol{s}$.

Заключение

Многие проблемы логического проектирования и других разделов информатики находят адекватное выражение на языке больших систем логических уравнений, решение которых представляет собой трудную комбинаторную задачу. В статье рассматриваются два класса таких систем, имеющие важные практические приложения и названные БСЛУ и СЛЛУ. Для решения систем данных классов была разработана серия комбинаторных методов и алгоритмов, описываемых в статье. Программная реализация и испытания на компьютере подтверждают высокую практическую эффективность этих методов и их применимость в широком диапазоне параметров решаемых систем: числа уравнений и переменных.

Список литературы

1. Теория дискретных управляющих устройств / Под ред. А.Д. Закревского и И.В. Прангишвили. – М.: Наука, 1982.

2. Лазарев В.Г., Пийль Е.И. Синтез управляющих автоматов. – М.: Энергия, 1970.

3. Закревский А.Д. Алгоритмы синтеза дискретных автоматов. – М.: Наука, 1971.

4. Основы технической диагностики / Под ред. П.П. Пархоменко. – М.: Энергия, 1976.

5. Чень Ч., Ли Р. Математическая логика и автоматическое доказательство теорем. – М.: Наука, 1983.

6. Математические и компьютерные основы криптологии / Ю.С. Харин, В.И. Берник, Г.В. Матвеев, С.В. Агиевич. – Мн.: ООО «Новое знание», 2003.

7. Алгоритмы решения логико-комбинаторных задач: Сб. науч. тр. / Под ред. А.Д. Закревского. – Мн.: Ин-т техн. кибернетики АН БССР, 1975-1980. – Вып. 1-6.

8. Закревский А.Д. Некоторые комбинаторные задачи искусственного интеллекта // Семиотика и информатика. – М.: ВИНИТИ, 1980. – Вып. 15. – С. 3-17.

9. Закревский А.Д. Комбинаторика логического проектирования // Автоматика и вычислительная техника. – 1990. – № 2. – С. 68-79.

10. Закревский А.Д. Комбинаторные задачи над логическими матрицами в логическом проектировании и искусственном интеллекте // Успехи современной радиоэлектроники. – 1998. – № 2. – С. 59-67.

11. Поттосин Ю.В. Задачи теории графов в логическом проектировании // Логическое проектирование. – Мн.: Ин-т техн. кибернетики НАН Беларуси, 2001. – Вып. 6. – С. 106-130.

12. Распознавание образов / Под ред. П. Колерса и М. Идена. – М.: Мир, 1970.

13. Закревский А.Д., Карелина А.В., Печерский Ю.Н. Всесоюзная школа-семинар по логико-комбинаторным методам в распознавании образов // Информационные материалы: Кибернетика. – М.: АН СССР, 1978. – № 4(104). – С. 16-18.

14. Zakrevskij A.D. A common logical approach to data mining and pattern recognition // Data Mining and Knowledge Discovery Approaches Based on Rule Induction Techniques. – Dordrecht: Kluwer Academic Publishers, 2004. – P. 1-42.

15. Рейнгольд Э., Нивергельт Ю., Део Н. Комбинаторные алгоритмы. Теория и практика. – М.: Мир, 1980.

16. Закревский А.Д. О приближенных методах решения логических задач // Вопросы синтеза цифровых автоматов. – М.: Наука, 1967. – С. 5-13.

17. Эйнгорин М.Я. О системах уравнений алгебры логики и синтезе дискретных управляющих схем с обратными связями // Известия вузов. Радиофизика. – 1958. – Т. 1. – № 2. – С. 169-184.

18. Rudeanu S. Boolean functions and equations. – Amsterdam-London-New York: North-Holland and American Elsevier, 1974.

19. Закревский А.Д. Логические уравнения. – Мн.: Наука и техника, 1975; М.: УРСС, 2003. – 2-е изд.

20. Уткин А.А. Решение логических уравнений // Автоматизация логического проектирования. – Мн.: Ин-т техн. кибернетики АН БССР, 1982. – С. 41-58.

21. Bochmann D., Zakrevskij A.D. Posthoff Ch. Boolesche Gleichungen. – Berlin: VEB Verlag Technik, 1984.

22. Закревский А.Д. Решение логических уравнений // Логическое проектирование. – Мн.: Ин-т техн. кибернетики НАН Беларуси, 2001. – Вып. 6. – С. 51-68.

23. Закревский А.Д. Логические уравнения с приложениями в автоматизированном проектировании и управлении // Автоматика и телемеханика, 2004. – № 4. – С. 173-185.

24. Закревский А.Д. Логический синтез каскадных схем. – М.: Наука, 1981. – 416 с.

25. Закревский А.Д. ПЛМ и матричные логические уравнения // Automatentheorie und Ihre Anwendung. Seminarbericht, Sekt. Math. der Humboldt-Universitet. – Berlin, 1984 Januar. – S. 26-34.

26. Бибило П.Н. Декомпозиция булевых функций на основе решения логических уравнений // Известия Академии наук. Теория и системы управления. – 2002. – № 4. – I. – С. 53-64; 2002. – № 5. – II. – С. 57-63; 2003. – № 6. – III. – С. 88-97.

27. Zakrevskij A.D. Pattern recognition as solving logical equations. – Special Issue 1999 – SSIT'99 (AMSE). – P. 125-136.

28. Лукасевич Я. Аристотелевская силлогистика с точки зрения современной формальной логики. – М.: ИЛ, 1959.

29. Закревский А.Д. К формализации полисиллогистики // Логический вывод. – М.: Наука, 1979. – С. 300-309.

30. Закревский А.Д. Матричный аппарат логического вывода в конечных предикатах // Философские основы неклассических логик: Тр. науч.-иссл. семинара по логике. – М.: Ин-т философии АН СССР, 1990. – С. 70-80.

31. Закревский А.Д. К решению систем логических уравнений // Принципы построения сетей и систем управления. – М.: Наука, 1964. – С. 48-55.

32. Закревский А.Д. Проверка тождеств в алгебре логики // Логический язык для представления алгоритмов синтеза релейных устройств. – М.: Наука, 1966. – С. 159-163.

33. Zakrevskii A.D., Kalmykova A.Yu. The solution of systems of logical equations // LYaPAS: A programming language for logic and coding algorithms – New-York and London: Academic Press, 1969. – P. 193-206.

34. Нильсон Н. Искусственный интеллект. Методы поиска решений. – М.: Мир, 1973.

35. Zakrevskij A., Zakrevski L. Solving systems of logical equations using search tree minimization technique // Proceedings of the PDPTA'02 International Conference, June 24-27, 2002, Las Vegas, USA. – P. 1145-1150.

36. Закревский А.Д., Василькова И.В. Решение больших систем логических уравнений: метод минимизации дерева поиска // Вестник Томского государственного университета. Приложение № 1 (II), сентябрь 2002. – С. 260-265.

37. Zakrevskij A., Vasilkova I. Reducing search trees to accelerate solving large systems of Boolean equations // Boolean Problems. 5th International Workshop, Sept. 19-20, 2002, Freiberg (Sachsen). – P. 71-76.

38. Закревский А.Д., Василькова И.В. Минимизация дерева поиска корней больших систем логических уравнений // Автоматика и вычислительная техника. – 2003. – № 1. – С. 3-11.

39. Cherry C., Vaswani P.K. A new type of computer in propositional logic, with greatly reduced scanning procedures // Information and control. – 1961. – V. 4. – № 3. – P. 155-168.

40. Zakrevskij A. Reduction algorithms for solving large systems of logical equations // Computer Science Journal of Moldova. – 2000. – V. 8. – № 1. – P. 3-15.

41. Zakrevskij A., Vasilkova I. Reducing large systems of Boolean equations // 4th International Workshop on Boolean Problems, September 21-22, 2000. – Freiberg, Germany. – P. 21-28.

42. Закревский А.Д., Василькова И.В. Криптоанализ машины Hagelin – метод решения системы логических уравнений // Комплексная защита информации. – Мн.: Ин-т техн. кибернетики АН Беларуси, 1999. – Вып. 2. – С. 129-138.

43. Закревский А.Д. Метод «отраженных волн» решения логических уравнений // Прикладные аспекты теории автоматов. Тр. III Междунар. семинара. Т. 1. – Варна: БАН, 1975. – С. 81-84.

44. Baumann M., Rohde R., Barthel R. Cryptoanalysis of the Hagelin M-209 Machine // 3rd International Workshop on Boolean Problems, Sept. 17-18, 1998. – Freiberg (Sachsen). – P. 109-116.

45. Закревский А.Д. Решение систем логических уравнений методом локальной редукции // Доклады НАН Беларуси, 1999. – Т. 43. – № 5. – С. 5-8.

46. Закревский А.Д., Василькова И.В. Локальная редукция больших систем логических уравнений // Логическое проектирование. – Мн: Ин-т техн. кибернетики АН Беларуси, 1999. – Вып.4.– С. 91-101.

47. Закревский А.Д. Редукция больших систем логических уравнений: метод силлогизмов // Автоматика и вычислительная техника. – 2000. – № 5. – С. 32-39.

48. Закревский А.Д., Торопов Н.Р. Полиномиальная реализация частичных булевых функций и систем. – Мн.: Ин-т техн. кибернетики НАН Беларуси, 2001. – 200 с.

49. Zakrevskij A.D. Looking for shortest solutions of systems of linear logical equations: theory and applications in logic design // 2 Workshop «Boolesche Probleme», 19 – 20 September 1996. Freiberg (Sachsen). – P. 63-69.

50. Закревский А.Д., Торопов Н.Р. Поиск кратчайшего решения системы линейных логических уравнений // Автоматизация проектирования дискретных систем: Мат. Второй междунар. конф. (CAD DD'97). – Мн.: Ин-т техн. кибернетики НАН Беларуси.– Т. 2. – С. 16-23.

51. Zakrevskij A.D., Zakrevski L. Optimizing solutions in a linear Boolean space – a decomposition method // Proc. of STI '2003, Orlando, Florida, USA, July 2003. – P. 276-280.

52. Закревский А.Д. Оптимизация решений в линейном булевом пространстве – методы декомпозиции // Автоматика и вычислительная техника. – 2003. – № 5. – С. 28-36.

53. Закревский А.Д. Комбинаторные методы оптимизации решений систем линейных логических уравнений // Вестник Томского государственного университета. Приложение. № 6. – Сентябрь 2003. – С. 4-8.

54. Закревский А.Д. Эффективные методы нахождения кратчайших решений систем линейных логических уравнений // Проблемы управления. – 2003. – № 4. – С. 16-22.

55. Zakrevskij A.D. Randomization of a parallel algorithm for solving undefined systems of linear logical equations // Proceedings of the International Workshop on Discrete-Event System Design – DESDes'04, 2004. – University of Zielona Gora Press, Poland. – P. 97-102.

56. Закревский А.Д., Василькова И.В. Прогнозирование затрат времени на реализацию комбинаторных алгоритмов // Методы логического проектирования. – Мн.: ОИПИ НАН Беларуси, 2003. – Вып. 2. – С. 26-32.

57. Закревский А.Д. Метод решения несовместных систем линейных логических уравнений // Вестник Томского государственного университета. Приложение. № 9(1). Сентябрь 2004. – С. 13-18.

58. Zakrevskij A.D. Solving inconsistent systems of linear logical equations // 6th International Workshop on Boolean Problems, September 23-24, 2004. Freiberg (Sachsen). – P. 183-190.

Поступила 30.10.04

Объединенный институт проблем
информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail:zakr@newman.bas-net.by

A.D. Zakrevskij

SOLVING LARGE SYSTEMS OF BOOLEAN EQUATIONS

Many problems of analysis, synthesis and diagnostics of logic circuits faults, as well as some problems of deductive inference, pattern recognition and information security are reduced to the solution of combinatorial tasks with inevitable exhaustive search of variants. A considerable proportion of such tasks is formulated in the language of logical equations. Two classes of systems containing hundreds of Boolean equations and variables are considered in this paper: LSLE (large systems of logical equations, with restricted number of variables in each of them) and SLLE (systems of linear logical equations). For solving such systems a series of combinatorial methods was developed in the Laboratory of Logical Design of the United Institute of Informatics Problems of NAS of Belarus. The results of the trials of these methods on the streams of pseudo-random systems testify to their high practical efficiency.