Серия «Математика»
2019. Т. 28. С. 3—20

Онлайн-доступ к журналу:
http://mathizv.isu.ru

ИЗВЕСТИЯ
Иркутского
государственного
университета

УДК 519.7
MSC 94C10
DOI https://doi.org/10.26516/1997-7670.2019.28.3

О способах пропозиционального кодирования различимости объектов в конечных множествах*

Е. Г. Белей

Иркутский национальный исследовательский технический университет, Иркутск, Российская Федерация

А. А. Семенов

Институт динамики систем и теории управления им. В. М. Матросова, Иркутск, Российская Федерация

Аннотация. В статье описана новая процедура пропозиционального кодирования свойства различимости объектов, составляющих некоторое конечное множество. Для изучаемого в настоящей работе класса комбинаторных проблем достаточно представлять элементы рассматриваемого множества их натуральными номерами. Соответственно, возникает задача построения выполнимой булевой формулы, выполняющие наборы которой кодируют первые n целых неотрицательных чисел без учета их порядка. Необходимость булевого кодирования таких множеств вызвана желанием применить к соответствующим комбинаторным задачам алгоритмы решения проблемы булевой выполнимости (SAT). В статье предлагается новый способ задания булевой формулой характеристической функции предиката, который истин на наборе двоичных слов, представляющих числа от 0 до $n-1$. Соответствующий предикат назван OtO (сокращение от One-to-One). Пропозициональная кодировка OtO-предиката имеет более экономную асимптотику роста в сравнении с кодировкой для близкого по смыслу предиката, известного как OOC-предикат (от Only One Cardinality). Описанный в статье OtO-предикат используется для сведения к SAT ряда задач, связанных с латинскими квадратами. Конкретно, с помощью OtO-предиката строятся SAT-кодировки для задач поиска ортогональных пар и квазиортогональных троек латинских квадратов порядка 10. Для вычислительного решения этих задач используются многопоточный SAT-решатель и вычислительный кластер.

Ключевые слова: комбинаторные объекты, латинские квадраты, пропозициональное кодирование, проблема выполнимости булевых формул, SAT.

* Работа выполнена при финансовой поддержке Российского научного фонда, грант 16–11–10046.

1. Введение

Про современную комбинаторику можно сказать, что это область, в которой встречаются друг с другом методы из, казалось бы, совершенно различных ветвей математики. Долгое время этот тезис относился, главным образом, к «чистой» математике. Датой начала проникновения в комбинаторику методов вычислительной математики можно считать 1960 год, когда Р. Ч. Боузом, С. С. Шрикханде и Е. Т. Паркером (см. [15]) была опровергнута гипотеза о греко-латинских квадратах, сформулированная Л. Эйлером в 1782 году.

На сегодня известно довольно много результатов, в рамках которых комбинаторные свойства доказываются с использованием компьютеров. Помимо упомянутой работы Боуза и др., еще один известный пример такого рода — доказательство К. Лэмом (см. [23]) отсутствия конечной проективной плоскости порядка 10. Вычислительный эксперимент, который провел Лэм, был настолько сложным, что соответствующая работа породила масштабную дискуссию о том, можно ли считать приведенные аргументы «доказательством» в общепринятом смысле. Буквально в последние несколько лет были построены вычислительные решения еще для нескольких комбинаторных проблем, относящихся к теории Рамсея. Так, в [22] было предъявлено вычислительное доказательство частного случая известной «Проблемы несоответствия Эрдёша» (Erdos discrepancy problem). Наконец, в [21] с использованием суперкомпьютера была решена еще одна известная комбинаторная задача – «Булева проблема пифагоровых троек» (Boolean Pythagorean triples problem).

В работах [21] и [22] к рассмотренным в них задачам применялись алгоритмы решения проблемы булевой выполнимости (Boolean Satisfiability Problem, SAT). Алгоритмы решения SAT в настоящее время используются в широком спектре областей: символьная верификация, биоинформатика, теория расписаний, криптоанализ и др. Следует отметить, что многие задачи из перечисленных областей допускают эффективную сводимость к SAT. Методы, обеспечивающие сводимости такого рода, называются процедурами пропозиционального кодирования.

В настоящей работе мы предлагаем новые процедуры пропозиционального кодирования некоторых комбинаторных объектов. Эти процедуры могут использоваться для решения различных задач, связанных с латинскими квадратами.

Приведем краткий план статьи. В разделе 2 вводятся основные понятия, относящиеся к проблематике булевой выполнимости. В разделе 3 мы приводим постановки ряда комбинаторных задач, в формулировках которых фигурируют латинские квадраты. В разделе 4 описаны общие принципы пропозиционального кодирования некоторых комбинаторных объектов. В разделе 5 приводится новая процедура пропо-

зиционального кодирования предиката над конечным множеством натуральных чисел, истинного только если все числа в рассматриваемом множестве различны. Мы используем эту схему кодирования для решения ряда комбинаторных задач, связанных с латинскими квадратами. Раздел 6 содержит результаты вычислительных экспериментов.

2. Задача о булевой выполнимости и основные алгоритмы ее решения

Переменные, принимающие значения в множестве из двух элементов, называются булевыми. Чаще всего значения булевых переменных обозначаются через 0 и 1. Булева формула от n переменных – это выражение, построенное по специальным правилам над алфавитом, который включает в себя булевы переменные $x_1, \ldots, x_n$, скобки и специальные символы, называемые булевыми (пропозициональными) связками. Для булевых формул также используются названия «пропозициональные формулы» или «формулы алгебры логики». Пусть $\{0,1\}^n$ — множество всех слов длины n над алфавитом $\{0,1\}$. В ряде источников элементы $\{0,1\}^n$ называются булевыми векторами. Любая булева формула F от n переменных задает всюду определенную (тотальную) булеву функцию $f_F : \{0,1\}^n \to \{0,1\}$. Формула F называется выполнимой, если найдется такой набор $\alpha \in \{0,1\}^n$ значений входящих в F переменных, что $f_F(\alpha) = 1$. Такой α называется набором, выполняющим F. Если выполняющего F набора не существует, то F называется невыполнимой. Выделяют булевы формулы специального вида, называемые нормальными формами (см., например, [8]). Основной объект дальнейшего рассмотрения — это конъюнктивная нормальная форма [8], при работе с которой используется аббревиатура КНФ (CNF). По любой булевой формуле F можно построить схему из функциональных элементов над произвольным полным базисом, например, над $\{\wedge, \neg\}$. По данной схеме, используя преобразования Цейтина [5], можно построить КНФ $C(F)$, которая выполнима тогда и только тогда, когда выполнима F. Формула $C(F)$ — это, вообще говоря, формула от большего числа переменных, чем F. Однако, что важно, переход от F к $C(F)$ осуществляется за полиномиальное время от длины двоичной записи F. Учитывая сказанное, везде далее под задачей о булевой выполнимости (сокращенно обозначаемой SAT) будем понимать проблему выполнимости произвольной КНФ: то есть по произвольной КНФ C требуется ответить на вопрос: «Верно ли, что C выполнима?» SAT – классическая NP-полная задача [17]. Таким образом, если $P \neq NP$, то SAT не может быть решена в общем случае за полиномиальное (от размера КНФ) время. Тем не менее в последние 20 лет наблюдается существенный прогресс в разработке алгоритмов решения SAT, показывающих впечатляющие результаты на обширных

классах так называемых индустриальных тестов. Начиная с 2001 года проводятся ежегодные соревнования программных реализаций алгоритмов решения SAT. Такие программы называются SAT-решателями (SAT-solvers). В основе подавляющего большинства современных полных SAT-решателей, эффективных на широких классах практических тестов, лежит алгоритм CDCL (Conflict Driven Clause Learning) [24]. В свою очередь данный алгоритм построен на основе известного еще с 60-х годов XX века алгоритма DPLL (от фамилий авторов: M. Davis, H. Putnam, G. Logemann, D. Loveland). Основное отличие между этими двумя алгоритмами заключается в том, что в CDCL используется память для хранения истории поиска — более точно, информация о тупиковых ветвях дерева поиска записывается в форме новых ограничений, называемых конфликтными дизъюнктами. Это позволяет в ряде случаев совершать существенно более глубокие откаты, чем в DPLL.

Алгоритм CDCL имеет экспоненциальную сложность на семействе булевых формул, известных как формулы Дирихле (Pigeon Hole Principle Formulas, PHP). Данный факт следует из полиномиальной эквивалентности CDCL и метода резолюций, что было установлено в [11], а также из экспоненциальной сложности метода резолюций на формулах Дирихле [20]. Несмотря на гарантированную неэффективность в худшем случае, современные SAT-решатели, основанные на CDCL, с успехом используются для решения задач из широкого спектра прикладных областей. Такие решатели оказываются эффективными даже в отношении задач криптоанализа: для целого ряда шифров их использование позволяет построить атаки, которые существенно эффективнее метода грубой силы [10; 26; 27] и др.

3. Задачи, связанные с латинскими квадратами

Латинский квадрат порядка n — это квадратная $n \times n$ таблица, заполненная символами произвольного алфавита $A = \{a_1, \ldots, a_n\}$ таким образом, что в каждой строке и каждом столбце встречаются все символы из A. Имеется тесная связь между латинскими квадратами и различными комбинаторными и алгебраическими структурами (квазигруппы, корректирующие коды, системы аутентификации, математические головоломки и многое другое). Целый ряд важных свойств латинских квадратов описан в классической монографии М. Холла [4]. Руководство [16] является, по-видимому, самым обширным источником информации о комбинаторных структурах в целом и о латинских квадратах в частности. Из недавних обзорных статей по комбинаторным структурам следует отметить работу [3], в которой представлен анализ свойств многомерных перманентов и их взаимосвязи с обширным классом комбинаторных структур, среди которых латинские гиперкубы,

являющиеся естественными обобщениями латинских квадратов. Помимо прочего, в данной работе приведены оценки числа трансверсалей в латинских квадратах, говорящие о том, что для малых порядков данное число может быть найдено точно в процессе вычислительного эксперимента. В [28] в результате такого эксперимента было найдено число трансверсалей в диагональных латинских квадратах порядка 9. В [29] был описан суперкомпьютерный эксперимент, результат которого — точное число диагональных латинских квадратов порядка 9 (ранее это комбинаторное число не было известно). Целый ряд близких по смыслу результатов может быть найден в статье [25].

Везде далее мы будем в роли алфавита A использовать множество натуральных чисел от 1 до n. Рассмотрим два латинских квадрата A, B порядка n. Занумеруем ячейки первого и второго квадратов в одинаковом порядке τ, используя для этой цели числа от 1 до n^2. Рассмотрим квадрат $A|B$, элементами которого являются все возможные пары (a, b), $a \in A, b \in B$, выписываемые в порядке τ (см. рисунок 1). Пары (a_i, b_i) и (a_j, b_j) различны по определению тогда и только тогда, когда $a_i \neq a_j$ или $b_i \neq b_j, i, j = 1, ..., n^2, i \neq j$.

Определение 1. *Квадрат $A|B$ называется греко-латинским, если все составляющие его пары различны. Если два латинских квадрата A, B образуют греко-латинский квадрат, то говорят, что A и B ортогональны (используется обозначение «$A \perp B$»).*

$$A=\begin{bmatrix}1 & 2 & 3\\ 2 & 3 & 1\\ 3 & 1 & 2\end{bmatrix} \quad B=\begin{bmatrix}1 & 2 & 3\\ 3 & 1 & 2\\ 2 & 3 & 1\end{bmatrix} \quad A|B=\begin{bmatrix}11 & 22 & 33\\ 23 & 31 & 12\\ 32 & 13 & 21\end{bmatrix}$$

Рис. 1. Пара ортогональных латинских квадратов порядка 3 и образованный ими греко-латинский квадрат.

Определение 2. *Набор из t латинских квадратов порядка n, в котором любые два квадрата ортогональны, называется системой из t ортогональных латинских квадратов порядка n или ортогональной системой мощности t и порядка n.*

Упомянутая выше гипотеза Эйлера предполагала, что не может существовать греко-латинских квадратов порядка $n = 4k+2$ для всех $k \geq 1$. Данная гипотеза была сформулирована в 1782 году и опровергнута Р. Ч. Боузом и С. С. Шрикханде в 1959 году в результате построения ортогональной пары порядка 22. Через год в совместной статье с Е. Т. Паркером они же построили первый пример ортогональной пары 10 порядка.

Еще в 1938 году в работе [14] Р. Ч. Боуз установил связь между системами ортогональных латинских квадратов и конечными проективными плоскостями. С одной стороны, можно показать, что если $L_1, \ldots, L_t$ — ортогональная система латинских квадратов порядка n, то $t \leq n-1$. С другой стороны, при $n \geq 3$ проективная плоскость порядка n существует тогда и только тогда, когда существует ортогональная система порядка n и мощности $n-1$.

Таким образом, проективная плоскость порядка 10 существует лишь тогда, когда существует ортогональная система, состоящая из 9 латинских квадратов порядка 10. Отсутствие такой системы, как уже говорилось выше, показал К. Лэм в 1991 году, проведя масштабный вычислительный эксперимент [23]. Возникает вопрос: «Какова наименьшая по мощности ортогональная система порядка 10?» Существует гипотеза, что это ортогональная система мощности 2, то есть ортогональная пара. Как было сказано выше, такие пары существуют (данный факт опровергает гипотезу Эйлера) и могут быть найдены различными способами. Таким образом, в соответствии с гипотезой, о которой идет речь, не может существовать ортогональной системы из 3 латинских квадратов порядка 10. Эта гипотеза не доказана и не опровергнута на сегодняшний день и является одной из наиболее известных открытых проблем конечной комбинаторики. Известно большое число попыток доказать данную гипотезу либо построить пример из трех попарно ортогональных латинских квадратов порядка 10. Однако все эти попытки окончились неудачей.

Далее будем называть гипотетическую систему из трех попарно ортогональных латинских квадратов порядка 10 ортогональной тройкой порядка 10. Отметим, что «приближаться» к построению такой системы можно последовательно. Для этого можно использовать следующие понятия (см., например, [16]). Пусть A, B — произвольные латинские квадраты порядка n. Рассмотрим квадрат $A|B$ и выпишем все пары вида (a_i, b_i), $i \in \{1, \ldots, n^2\}$, в некотором фиксированном порядке. Пусть r — число различных пар в полученной последовательности (в указанном выше смысле).

Определение 3. *Число $r = r(A, B)$, $1 \leq r \leq n^2$, называется индексом ортогональности пары A, B, а сама эта пара называется квазиортогональной парой индекса r.*

Пусть A, B — латинские квадраты порядка n. Очевидно, что $A \perp B$ тогда и только тогда, когда A, B – квазиортогональная пара индекса n^2.

Определение 4. *Пусть L — множество из t латинских квадратов порядка n. Пусть r — минимальное значение индекса ортогональности по всем парам квадратов из L. Скажем, что L — квазиортогональная система индекса r.*

Насколько нам известно, одной из наиболее близких по индексу ортогональности к ортогональной тройке порядка 10 является система, приведенная в работе [19]. Конкретно это система из трех латинских квадратов L_1, L_2, L_2: $L_1 \perp L_2$, $L_1 \perp L_3$, однако (L_2, L_3) — квазиортогональная пара индекса 91.

4. Общие принципы пропозиционального кодирования комбинаторных задач, связанных с латинскими квадратами

В данном разделе речь пойдет об общих принципах сведения различных задач о существовании комбинаторных структур к задаче о булевой выполнимости. Основным объектом нашего интереса будут задачи, связанные с латинскими квадратами.

Неформально все дальнейшие построения основаны на идее использовать булевы формулы для задания следующего свойства: «Все элементы в некотором n-элементном множестве различны». Можно ограничиться натуральными номерами элементов и тогда фактически требуется выразить булевой формулой свойство, состоящее в том, что рассматриваемое множество образовано первыми n натуральными числами с точностью до их порядка.

Поясним связь данной задачи с задачами существования комбинаторных объектов. Будем действовать «поступательно», определяя сложные комбинаторные объекты на основе более простых. Элементарный комбинаторный объект в нашем случае — ячейка. Будем полагать, что ячейка может обладать любым свойством из некоторого конечного множества. Удобнее всего представлять данные свойства натуральными числами. Если рассматривается система из нескольких различных ячеек, то каждая такая ячейка также может быть занумерована натуральным числом. Таким образом, имеется два набора натуральных чисел: один набор фиксирован и задает порядок на множестве ячеек, другой — это алфавит $A = \{1, \ldots, n\}$ элементы которого могут быть записаны в ячейки произвольным образом. Наборы ячеек могут образовывать более сложные комбинаторные объекты, например, строка из n ячеек — это также комбинаторный объект. Наиболее интересное для нас свойство строки заключается в том, что все числа в ячейках, формирующих эту строку, различны. Из n строк, действуя по аналогии, мы определяем латинский квадрат. Затем пары ортогональных латинских квадратов и т. д.

Вычислительные алгоритмы, которые мы хотим использовать для поиска тех или иных комбинаторных объектов (например, латинских квадратов), должны выдавать описания этих объектов в виде двоичных векторов, являющихся выполняющими наборами некоторых выполни-

мых формул. Либо, если искомого объекта не существует, соответствующая формула должна быть невыполнимой.

Обозначим через $sol\,(F)$ множество наборов, выполняющих произвольную булеву формулу F. Базовая идея пропозиционального кодирования состоит в следующем. Пусть имеется описание W некоторого комбинаторного объекта. Данному описанию ставится в соответствие булева формула F, такая, что $sol(F) = \emptyset$ тогда и только тогда, когда не существует комбинаторных объектов удовлетворяющих описанию W. В противном случае должно существовать тотальное сюръективное отображение множества $sol(F)$ на множество объектов, удовлетворяющих W. Произвольное $\alpha \in sol(F)$ назовем пропозициональным кодом комбинаторного объекта, удовлетворяющего W. Процесс построения формулы F называется пропозициональным кодированием.

Поясним приведенные выше понятия и принципы на примере построения семейства выполнимых булевых формул F_n, $n \in N$, такого, что для конкретного $n \geq 2$ существует сюръекция $sol\,(F_n) \to LS_n$, где через LS_n обозначено множество латинских квадратов порядка n.

Рассмотрим множество ячеек занумерованных числами от 1 до n. Обозначим данное множество через $O = \{o_1, \ldots, o_n\}$. Рассмотрим алфавит $A{=}\{1, \ldots, n\}$. Назовем конфигурацией типа 0 произвольную пару $(o, a) \in O \times A$. Рассмотрим следующее описание $W =$ «$\aleph$ — такой набор комбинаторных конфигураций типа 0, что задействованными оказываются все ячейки из O, и любым двум ячейкам, входящим в $\aleph$, приписаны различные буквы из A». Произвольную конфигурацию, удовлетворяющую описанию W, назовем комбинаторной конфигурацией типа 1.

Опишем теперь известный способ пропозиционального кодирования комбинаторных конфигураций первого и второго типов. Итак, пусть $O = \{o_1, \ldots, o_n\}$ — множество из n ячеек. Сопоставим произвольному o_i, $i \in \{1, \ldots, n\}$, множество булевых переменных $X^i = \left\{x_1^i, \ldots, x_n^i\right\}$. Рассмотрим следующую булеву формулу:

$$F_{OOC}\left(X^i\right) = \left(x_1^i \vee \cdots \vee x_n^i\right) \wedge \bigwedge\nolimits_{j,k \in \{1,\ldots,n\}: j \neq k} \left(\neg x_j^i \vee \neg x_k^i\right). \qquad (4.1)$$

Несложно понять, что $sol\left(F_{OOC}(X^i)\right)$ образован всеми различными булевыми векторами длины n, вес Хэмминга которых равен 1. Тем самым каждый вектор $\alpha \in sol\left(F_{OOC}(X^i)\right)$ мы можем рассматривать как двоичный код комбинаторной конфигурации типа 0: ячейка, с номером i заполнена числом k, где k — номер позиции единичного бита в α.

Аббревиатура OOC — это сокращение от Only One Cardinality (см., например, [9]). Очевидно, что формулу 4.1 можно рассматривать как способ задания свойства (предиката) булевого вектора длины n иметь единственный единичный бит. Характеристическая функция данного предиката определена всюду на $\{0,1\}^n$, принимает значение «истина»

только на n единичных векторах и принимает значение «ложь» на всех остальных векторах.

Введем n множеств вида $X^i = \{x_1^i, \ldots, x_n^i\}$, $i \in \{1, \ldots, n\}$, организуем их в виде $n \times n$ матрицы, столбец с номером $i, i \in \{1, \ldots, n\}$, которой составлен из переменных $x_1^i, \ldots, x_n^i$. Пусть $\tilde{X}^l$ – множество переменных, образующих строку данной матрицы с номером l, $l \in \{1, \ldots, n\}$, то есть $\tilde{X}^l = \{x_l^1, \ldots, x_l^n\}$. Рассмотрим следующую формулу:

$$F_{OOC}\left(X^1\right) \wedge \cdots \wedge F_{OOC}\left(X^n\right) \wedge F_{OOC}\left(\tilde{X}^1\right) \wedge \cdots \wedge F_{OOC}\left(\tilde{X}^n\right). \quad (4.2)$$

Несложно понять, что произвольный набор значений переменных из $\bigcup_{i=1}^{n} X^i$, выполняющий 4.2, если его разбить на участки (сегменты), соответствующие множествам X^i, $i \in \{1, \ldots, n\}$, содержит все n различных единичных векторов из $\{0,1\}^n$ и, таким образом, кодирует комбинаторную конфигурацию типа 1.

Рассмотрим теперь множество L_n, образованное n^2 ячейками. Будем рассматривать L_n как квадратную матрицу, каждая строка и каждый столбец которой содержат n ячеек, при этом каждой ячейке в L_n приписан собственный уникальный номер и произвольное число из множества $\{1, \ldots, n\}$. Очевидно, что L_n является латинским квадратом порядка n тогда и только тогда, когда каждая строка и каждый столбец L_n образуют комбинаторную конфигурацию типа 1. Произвольный латинский квадрат будем называть комбинаторной конфигурацией типа 2. Легко видеть, что в пропозициональной кодировке латинского квадрата порядка n, основанной на OOC-предикате, число переменных растет с ростом n как n^3, а число дизъюнктов как $\Theta\left(n^4\right)$.

5. Новый алгоритм пропозиционального кодирования различимости объектов в конечном множестве

В данном разделе мы описываем схему пропозиционального кодирования комбинаторных конфигураций типов 0-2, которая базируется на технике, не использующей OOC-предикат.

Заметим, что довольно очевидной альтернативой рассмотренного выше способа описания произвольного натурального числа $i \in \{1, \ldots, n\}$ единичными вектором длины n является описание данного числа вектором длины $\lceil \log n \rceil$, являющимся двоичным представлением числа $i-1$. Более точно, произвольному числу $m \in \{0, 1, \ldots, n-1\}$ сопоставим булев вектор $\alpha = \left(\alpha_1, \ldots, \alpha_{\lceil \log n \rceil}\right)$, являющийся вектором коэффициентов его двоичного представления:

$$\begin{gathered} m = \alpha_1 + \alpha_2 \cdot 2 + \cdots + \alpha_{\lceil \log n \rceil} \cdot 2^{\lceil \log n \rceil - 1}, \\ \alpha_i \in \{0,1\}, i \in \{1, \ldots, \lceil \log n \rceil\}. \end{gathered} \quad (5.1)$$

Рассмотрим множество $\{0,1\}^{n\cdot\lceil\log n\rceil}$. Разобьем произвольное слово $\gamma \in \{0,1\}^{n\cdot\lceil\log n\rceil}$ на n последовательных подслов длины $\lceil\log n\rceil$ каждое. Обозначим полученное множество через Λ_γ. Будем говорить, что множество Λ_γ порождено словом γ. Определим предикат

$$OtO : \{0,1\}^{n\cdot\lceil\log n\rceil} \to \{False, True\}$$

как свойство произвольного слова γ длины $n \cdot \lceil\log n\rceil$ порождать множество Λ_γ, содержащее расположенные в произвольном порядке векторы двоичных представлений (в соответствии с (5.1)) всех чисел из множества $\{0,1,\ldots,n-1\}$.

Определение 5. *Про произвольное $\gamma \in \{0,1\}^{n\cdot\lceil\log n\rceil}$, на котором OtO принимает значение True, будем говорить, что данное слово обладает OtO-свойством.*

Очевидно, что произвольное слово γ из $\{0,1\}^{n\cdot\lceil\log n\rceil}$, на котором OtO-предикат принимает значение True, можно рассматривать как двоичный код комбинаторной конфигурации типа 1. В этом случае будем также говорить, что γ задает такую конфигурацию.

Построим теперь булеву формулу, задающую OtO-предикат. С этой целью будем рассматривать нашу базовую задачу: имеется множество ячеек-объектов $O = \{o_1,\ldots,o_n\}$ и множество чисел $P = \{1,\ldots,n\}$.

Свяжем с каждым объектом $o_i \in O$, $i \in \{1,\ldots,n\}$, множество, состоящее из $s = \lceil\log n\rceil$ булевых переменных $X^i = \left\{x_1^i,\ldots,x_s^i\right\}$. Для произвольного $m \in \{0,1,\ldots,n-1\}$ обозначим через

$$(\alpha_1(m),\ldots,\alpha_s(m))$$

вектор двоичного представления 5.1 числа m. Везде далее запись x^λ как обычно (см. [8]) означает литерал $\neg x$ при $\lambda = 0$, и литерал x при $\lambda = 1$.

Введем в рассмотрение следующую формулу:

$$\phi\left(X^i, m\right) = \left(x_1^i\right)^{\alpha_1(m)} \wedge \cdots \wedge \left(x_s^i\right)^{\alpha_s(m)},$$

Таким образом, $\phi\left(X^i, m\right)$ принимает значение 1 только на наборе значений переменных X^i, который является вектором коэффициентов двоичного представления 5.1 числа m.

Рассмотрим формулу:

$$\Phi(X) = \bigwedge_{i=1}^{n} \bigvee_{m=0}^{n-1} \left(\left(x_1^i\right)^{\alpha_1(m)} \wedge \cdots \wedge \left(x_s^i\right)^{\alpha_s(m)}\right). \tag{5.2}$$

Лемма 1. *Формула $\Phi(X)$ задает OtO-предикат.*

Доказательство. Запишем 5.2 в следующем виде:

$$\Phi(X) = \bigwedge_{i=1}^{n} \Big(\big(x_1^i\big)^{\alpha_1(0)} \wedge \cdots \wedge \big(x_s^i\big)^{\alpha_s(0)} \vee \ldots \\ \vee \big(x_1^i\big)^{\alpha_1(n-1)} \wedge \cdots \wedge \big(x_s^i\big)^{\alpha_s(n-1)} \Big). \tag{5.3}$$

Рассмотрим произвольный $\gamma \in \{0,1\}^{n\cdot\lceil \log n \rceil}$. Разобьем его на n последовательных сегментов, рассматривая данный вектор как набор значений переменных из $X^1, \ldots, X^n$. Предположим, что γ состоит из двоичных представлений всех чисел от 0 до $n-1$, записанных в произвольном порядке. Тогда для каждого $i \in \{1, \ldots, n\}$ найдется такое $m \in \{0, \ldots, n-1\}$ что формула вида $\phi(X^i, m)$ выполнена на векторе γ. Соответственно, $\Phi(X)|_\gamma = 1$.

Пусть теперь γ – произвольный вектор из $\{0,1\}^{n\cdot\lceil \log n \rceil}$, не обладающий OtO-свойством. Легко понять, что в данном случае найдется такое $i \in \{1, \ldots, n\}$, что для любого $m \in \{0, \ldots, n-1\}$ формула $\phi\left(X^i, m\right)$ примет значение 0 на γ. Тем самым имеет место $\Phi(X)|_\gamma = 0$. Лемма 1 доказана. □

Далее мы покажем, что пропозициональная кодировка конфигурации типа 1, построенная на основе OtO-предиката, является асимптотически существенно более экономной, чем на основе OOC-предиката, по числу дизъюнктов. Также мы увидим, что кодировка на основе OtO выгодно отличается от OOC-кодировки в смысле еще одного свойства. Для понимания его сути нам потребуется понятие так называемых лазеек (Backdoor sets), введенное в [30].

Рассмотрим произвольную КНФ C над множеством переменных X. Для произвольного $X', X' \subseteq X$, через $\{0,1\}^{|X'|}$ обозначим множество наборов значений переменных, входящих в X'. Через $C\left[X'/\beta\right]$, $\beta \in \{0,1\}^{|X'|}$ обозначим КНФ, которая получается из C в результате подстановки набора β значений переменных из X' (подстановка определяется стандартным способом — см., например, [7]).

Определение 6. *(см. [30]). Пусть A — некоторый полиномиальный алгоритм. Если для любого $\beta \in \{0,1\}^{|X'|}$ алгоритм A выдает верный ответ на вопрос о выполнимости КНФ $C\left[X'/\beta\right]$, то X' называется лазейкой (Backdoor Set) для КНФ C относительно алгоритма A.*

Применение лазеек часто позволяет определить выполнимость КНФ существенно быстрее, чем без их использования. Очевидно, что SAT в отношении КНФ C с лазейкой X' можно решать в параллельном режиме, рассматривая $2^{|X'|}$ SAT-задач в отношении КНФ вида $C\left[X'/\beta\right]$ и решая каждую из них алгоритмом A.

Определение 7. *Если алгоритм A – это правило единичного дизъюнкта (Unit Propagation rule, UP [24]), то лазейку для КНФ C относительно A будем называть UP-лазейкой.*

UP-лазейки играют важную роль при использовании SAT для обращения криптографических функций. Знание UP-лазеек в таких случаях позволяет строить атаки, существенно более эффективные, чем метод грубой силы (см., например, [27]). При этом UP-лазейки меньшей мощности дают более эффективные атаки. Основным результатом данного пункта является следующая теорема.

Теорема 1. *Существует пропозициональная кодировка латинского квадрата L_n, основанная на OtO-предикате, с асимптотикой роста числа переменных $\Theta\left(n^3\right)$ и асимптотикой роста числа дизъюнктов $\Theta\left(n^3 \cdot \log n\right)$, имеющая UP-лазейку мощности $n^2 \cdot \lceil \log n \rceil$.*

Доказательство. Итак, представим OtO-предикат формулой 5.3. Наша цель — перейти от задачи выполнимости 5.3 к задаче выполнимости КНФ. Используем для этого преобразования Цейтина [5]. Для каждого $m \in \{0, \ldots, n-1\}$ и произвольного $i \in \{1, \ldots, n\}$ введем новую переменную z_i^m и рассмотрим формулу

$$z_i^m \equiv \left(x_1^i\right)^{\alpha_1(m)} \wedge \cdots \wedge \left(x_s^i\right)^{\alpha_s(m)}. \tag{5.4}$$

КНФ-представление формулы 5.4 содержит s+1 дизъюнкт. Обозначим эту КНФ через ψ_i^m. Рассмотрим следующую КНФ:

$$\left(\bigwedge\nolimits_{i=1}^{n}\left(z_i^0 \vee \cdots \vee z_i^{n-1}\right)\right) \wedge \bigwedge\nolimits_{m=0}^{n-1}\bigwedge\nolimits_{i=1}^{n}\psi_i^m. \tag{5.5}$$

В силу свойств преобразований Цейтина существует биекция между множеством наборов значений переменных из X, выполняющих 5.2, и множеством наборов перемененных из $X \bigcup \{z_i^m\}$, $i \in \{1, \ldots, n\}$, $m \in \{0, \ldots, n-1\}$, выполняющих 5.5. Тем самым мы свели задачу выполнимости формул вида $\Phi(X)$ к задаче выполнимости КНФ вида 5.5 с асимптотикой по переменным $\Theta\left(n^2\right)$ и асимптотикой по дизъюнктам $\Theta\left(n^2 \cdot \log n\right)$. Данная КНФ кодирует комбинаторную конфигурацию типа 1 (например, произвольную строку латинского квадрата L_n). Соответственно, для КНФ, кодирующей весь квадрат L_n целиком, будет иметь место асимптотика по переменным $\Theta\left(n^3\right)$ и асимптотика по дизъюнктам $\Theta\left(n^3 \cdot \log n\right)$.

С формулами, кодирующими латинский квадрат порядка n, построенными на основе OOC-предиката и на основе OtO-предиката, естественным образом связываются функции вида $f_{OOC} : \{0,1\}^{n^3} \to \{0,1\}$ и $f_{OtO} : \{0,1\}^{n^2 \cdot \lceil \log n \rceil} \to \{0,1\}$. Например, в случае функции f_{OOC} множество $\{0,1\}^{n^3}$ рассматривается как множество значений переменных

из $X = \bigcup_{i=1}^{n^2} X^i$, приписанных ячейкам латинского квадрата: f_{OOC} принимает значение 1 на любом таком векторе $\alpha \in \{0,1\}^{n^3}$, для которого выполнены все OOC-условия, задающие строки и столбцы данного латинского квадрата, и принимает значение 0 на всех остальных векторах из $\{0,1\}^{n^3}$. Аналогичным образом определяется функция f_{OtO}. Функции f_{OOC} и f_{OtO} можно задать схемами из функциональных элементов S_{OOC} и S_{OtO} (соответственно) над базисом $\{\wedge, \neg\}$. Описанные выше пропозициональные кодировки латинского квадрата L_n могут быть эффективно построены в результате преобразований Цейтина, примененных к схемам S_{OOC} и S_{OtO}. В работах [2;12] был установлен следующий факт. Пусть $f : \{0,1\}^M \to \{0,1\}^N$ — произвольная дискретная функция, заданная схемой $S(f)$ из функциональных элементов над базисом $\{\wedge, \neg\}$, и $C(f)$ — КНФ, построенная по схеме $S(f)$ в результате преобразований Цейтина. Тогда множество X^{in} переменных в $C(f)$, приписанных входным полюсам схемы $S(f)$, является UP-лазейкой для $C(f)$. Из сказанного выше следует, что в случае функции f_{OOC} мощность X^{in} есть n^3, а в случае f_{OtO} множество X^{in} имеет мощность $n^2 \cdot \lceil \log n \rceil$. Теорема 1 доказана. □

Отметим, что описанная выше кодировка латинского квадрата на основе OtO-предиката асимптотически существенно экономнее, чем OOC-кодировка по дизъюнктам и имеет меньшую UP-лазейку. Данные кодировки имеют одинаковую асимптотику по переменным. Однако на самом деле число переменных в OtO-кодировке больше, поскольку основной вклад в асимптотику $\Theta\left(n^3\right)$ вносят «цейтиновские» переменные вида z_i^m, притом что помимо них в кодировке присутствуют и $n^2 \cdot \lceil \log n \rceil$ переменных, кодирующих содержимое ячеек квадрата.

6. Вычислительные эксперименты

В данном разделе мы приводим результаты использования пропозициональной кодировки OtO-предиката, описанной в теореме 1, для поиска некоторых комбинаторных структур, в основе которых лежат латинские квадраты. Мы реализовали алгоритм пропозиционального кодирования латинских квадратов из теоремы 1, а также кодировку на основе OOC-предиката. Соответствующие алгоритмы строят КНФ в формате DIMACS [18]. К этим КНФ мы применяли известный многопоточный SAT-решатель Plingeling [13]. Все эксперименты проводились на одном узле вычислительного кластера Иркутского суперкомпьютерного центра «Академик В.М. Матросов» [6].

Первая серия экспериментов относится к поиску ортогональных пар латинских квадратов порядка 10 (каждая такая пара опровергает гипотезу Эйлера). Решатель Plingeling — рандомизированный алгоритм.

С учетом этого мы сделали по 10 независимых запусков Plingeling на OOC- и OtO-кодировках, кодирующих существование ортогональных пар латинских квадратов порядка 10. По результатам данных экспериментов OtO-кодировка оказалась в среднем в два раза эффективнее OOC-кодировки (среднее время нахождения одной ортогональной пары порядка 10 при использовании OtO-кодировки составило менее 15 минут). Вторая серия экспериментов заключалась в поиске квазиортогональных троек латинских квадратов порядка 10. Как было сказано в пункте 3, гипотетическая ортогональная тройка порядка 10 — это на самом деле квазиортогональная система из трех латинских квадратов с индексом ортогональности 100. При кодировании условий на индекс ортогональности мы использовали методы, описанные в статье [1]. Как результат, были найдены квазиортогональные тройки порядка 10 с индексом ортогональности до 85 включительно. На данной серии экспериментов OtO-кодировка проиграла по эффективности OOC-кодировке в среднем около 10%.

7. Заключение

В настоящей статье мы описали новый алгоритм пропозиционального кодирования предиката над n натуральными числами, истинного в том случае, если все эти числа различны. Мы использовали построенную кодировку для сведения некоторых комбинаторных задач, связанных с латинскими квадратами, к проблеме булевой выполнимости (SAT). К получаемым SAT-задачам применялся многопоточный решатель Plingeling, с использованием которого были проведены эксперименты на вычислительном кластере «Академик В. М. Матросов» Иркутского суперкомпьютерного центра. На некоторых задачах новая кодировка по эффективности существенно превзошла известную схему кодирования, основанную на OOC-предикате.

Список литературы

1. Белей Е. Г., Семенов А. А. О вычислительном поиске квазиортогональных систем латинских квадратов, близких к ортогональным системам // International Journal of Open Information Technologies. 2018. Vol. 6, N 2, P. 22-30.
2. Семенов А. А. Декомпозиционные представления логических уравнений в задачах обращения дискретных функций // Изв. РАН. 2009. № 5. С. 47–61.
3. Тараненко А. А. Перманенты многомерных матриц: свойства и приложения // Дискретный анализ и исследование операций. 2016. Т. 23, № 4. С. 35-101. https://doi.org/10.17377/daio.2016.23.517
4. Холл М. Комбинаторика. М. : Мир, 1970. 424 с.

5. Цейтин Г.С. О сложности вывода в исчислении высказываний // Записки научных семинаров ЛОМИ. 1968. Т. 8. С. 234–259.
6. ЦКП Иркутский суперкомпьютерный СО РАН [Электронный ресурс]. http://hpc.icc.ru (дата обращения: 28.04.2019).
7. Чень Ч., Ли Р. Математическая логика и автоматическое доказательство теорем. М. : Наука, 1983. 360 с.
8. Яблонский С. В. Введение в дискретную математику. М. : Наука, 1986. 384 с.
9. Asin R., Nieuwenhuis R., Oliveras A., Rodriguez-Carbonell E. Cardinality networks: a theoretical and empirical study // Constraints. Vol. 16. Iss. 2011. P. 195–221. https://doi.org/10.1007/s10601-010-9105-0
10. Bard G. V. Algebraic Cryptanalysis. Springer Publishing Company, Incorporated, last Edition, 2009. https://doi.org/10.1007/978-0-387-88757-9
11. Beame P., Kautz H., Sabharwal A. Understanding the power of clause learning // IJCAI (18th International Joint Conference on Artificial Intelligence). 2003. P. 1194–1201.
12. Bessiere C., Katsirelos G., Narodytska N., Walsh T. Circuit complexity and decompositions of global constraints. In IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence. 2009. P. 412–418.
13. Biere A. Splatz, Lingeling, Plingeling, Treengeling, YalSAT Entering the SAT Competition 2016 // Proc. of SAT Competition / Balyo T., Heule M., Jarvisalo M. (eds.). 2016. P. 44–45.
14. Bose R. C. On the application of the properties of Galois fields to the problem of construction of hyper-Graeco-Latin squares // Sankhya. 1938. Vol. 3. P. 323–338.
15. Bose R. C., Shrikhande S. S., Parker E. T. Further results on the construction of mutually orthogonal Latin squares and the falsity of Euler's conjecture // Canadian Journal of Mathematics. 1960. P. 189-203.
16. Colbourn C. J., Dinitz J. H. Handbook of combinatorial designs. Second edition. CRC Press, 2010. http://doi.org/10.1201/9781420010541
17. Cook S. A. The complexity of theorem-proving procedures // Third annual ACM symposium on Theory of computing, 1971, Ohio, USA. P. 151–159.
18. DIMACS Implementation Challenges [Electronic resource]. URL: http://dimacs.rutgers.edu/Challenges/
19. Egan J., Wanless I. M., Enumeration of MOLS of small order // Math. Comput. 2016. Vol. 85. P. 799–824. https://doi.org/10.1090/mcom/3010
20. Haken A. The intractability or resolution // Theoretical Computer Science. 1985. Vol. 39. P. 297–308.
21. Heule M. J. H., Kullman O., Marec V. W. Solving and Verifying the Boolean Pythagorean Triples Problem via Cube-and-Conquer // Lecture Notes in Computer Science. 2014. Vol. 9710 (SAT-2016). P. 228–245.
22. Konev B., Lisitsa A. A SAT Attack on Erdos Discrepancy Problem // Lecture Notes in Computer Science. 2014. Vol. 8561 (SAT-2014). P. 219–226. https://doi.org/10.1007/978-3-319-09284-3_17
23. Lam C.W.H. The Search for a Finite Projective Plane of Order 10. The American Mathematical Monthly. 1991. P. 305-318.
24. Marques-Silva J. P., Lynce I., Malik S. Conflict-Driven Clause Learning SAT solvers // Handbook of Satisfiability / Biere A., Heule M., van Maaren H., Walsh T. (eds.). 2009. P. 131–153. https://www.doi.org/10.3233/978-1-58603-929-5-131
25. McKay B. D., Wanless I. M. A census of small Latin hypercubes // SIAM J. Discrete Math. 2008. Vol. 22. P. 719–736. https://doi.org/10.1137/070693874
26. Mironov I., Zhang L. Applications of SAT solvers to cryptanalysis of hash functions // Lecture Notes in Computer Science. 2006. Vol. 4121. P. 102-115. https://doi.org/10.1007/11814948_13

27. Semenov A., Zaikin O., Otpuschennikov I., Kochemazov S., Ignatiev A. On cryptographic attacks using backdoors for SAT. In The Thirty-Second AAAI Conference on Artificial Intelligence, AAAI'2018, pages 6641-6648, 2018.
28. Vatutin E., Kochemazov S., Zaikin O., Valyaev S. Enumerating the Transversals for Diagonal Latin Squares of Small Order // Ceur-WS. 2017. Vol. 1973 (BOINC:FAST 2017). P. 6–14.
29. Vatutin E., Kochemazov S., Zaikin O. Applying Volunteer and Parallel Computing for Enumerating Diagonal Latin Squares of Order 9// Communication in Computer and Information Science. 2017. Vol. 753. P. 114–129. https://doi.org/10.1007/978-3-319-67035-5_9
30. Williams R., Gomes C. P., Selman B. Backdoors to typical case complexity // In 18th International Joint Conference on Articial Intelligence - IJCAI 2003. 2003. P. 1173-1178.

Евгения Геннадьевна Белей, магистрант, Иркутский национальный исследовательский технический университет, Российская Федерация, 664074, г. Иркутск, ул. Лермонтова, 83, тел.: (908)6684039
(e-mail: zhenyha28@mail.ru)

Александр Анатольевич Семенов, кандидат технических наук, доцент, зав. лабораторией, Институт динамики систем и теории управления им. В. М. Матросова СО РАН, Российская Федерация, 664033, г. Иркутск, ул. Лермонтова, 134, тел.: (914)8707647
(e-mail: biclop.rambler@yandex.ru)

Поступила в редакцию 30.04.19

On Propositional Encoding of Distinction Property in Finite Sets

E. G. Beley

Irkutsk National Research Technical University, Irkutsk, Russian Federation

A. A. Semenov

Matrosov Institute for System Dynamics and Control Theory of SB RAS, Irkutsk, Russian Federation

Abstract. In the paper we describe a new propositional encoding procedure for the property that all objects comprising some finite set are distinct. For the considered class of combinatorial problems it is sufficient to represent the elements of such set by their natural numbers. Thus, there is a problem of constructing a satisfiable Boolean formula, the satisfying assignments of which encode the first n natural numbers without taking into account their order. The necessity to encode such sets into Boolean logic is motivated by the desire to apply to the corresponding combinatorial problems the state-of-the-art algorithms for solving the Boolean satisfiability problem (SAT). In the paper we propose the new approach to defining the Boolean formula for the characteristic function of the predicate which takes the value of True only on the sets of binary words encoding the numbers from 0 to $n-1$. The corresponding predicate was named OtO (from One-

to-One). The propositional encoding of the OtO-predicate has a better lower bound compared to the propositional encoding of a relatively similar predicate known as OOC-predicate (from Only One Cardinality). The proposed OtO-predicate is used to reduce to SAT a number of problems related to Latin squares. In particular, using the OtO-predicate we constructed the SAT encodings for the problems of finding orthogonal pairs and quasi-orthogonal triples of Latin squares of order 10. We used the multi-threaded SAT solvers and the resources of a computing cluster to solve these problems.

Keywords: propositional encoding, SAT, latin squares.

References

1. Beley E.G., Semenov A.A. On computational search for quasi-orthogonal systems of Latin squares which are close to orthogonal system. *International Journal of Open Information Technologies*, 2018, vol. 6, no. 2, pp. 22-30. (in Russian)
2. Semenov A.A. Decomposition representations of logical equations in inversion problems of discrete functions. *News of the Russian Academy of Sciences*, 2009, no. 5, pp. 47-61. (in Russian)
3. Taranenko A.A. Permanents of multidimensional matrices: Properties and applications. *Journal of Applied and Industrial Mathematics*, 2016, vol. 10, no. 4, pp. 567-604. https://doi.org/10.1134/S1990478916040141
4. Hall M. Combinatorics. Moscow, Mir Publ., 1970. 424 p.
5. Tseitin G.S. About the complexity of the conclusion in the calculus of statements. *Notes of scientific seminars LOMI*, 1968, vol. 8., pp.234-259. (in Russian)
6. CCU Irkutsk supercomputer SB RAS [Electronic resource]. URL: http://hpc.icc.ru (the date of handling: 04.28.2019).
7. Chen C., Lee R. Mathematical logic and automatic proof of theorems. Moscow, Nauka Publ. 1983. 360 p. (in Russian)
8. Yablonsky S.V. Introduction to discrete mathematics. Moscow, Nauka Publ. 1986. 384 p.(in Russian)
9. Asin R., Nieuwenhuis R., Oliveras A., Rodriguez-Carbonell E. Cardinality networks: a theoretical and empirical study. *Constraints*, vol. 16, Iss. 2011, pp. 195–221. https://doi.org/10.1007/s10601-010-9105-0
10. Bard G.V. *Algebraic Cryptanalysis*. Springer Publishing Company, Incorporated, 2009. https://doi.org/10.1007/978-0-387-88757-9
11. Beame P., Kautz H., Sabharwal A. Understanding the power of clause learning. *Proceedings of the18th International Joint Conference on Artificial Intelligence)*, 2003, pp. 1194-1201.
12. Bessiere C., Katsirelos G., Narodytska N., Walsh T. Circuit complexity and decompositions of global constraints. *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, 2009, pp. 412-418.
13. Biere A. Splatz, Lingeling, Plingeling, Treengeling, YalSAT Entering the SAT Competition 2016. *Proc. of SAT Competition 2016*, pp. 44–45.
14. Bose R.C. On the application of the properties of Galois fields to the problem of construction of hyper-Graeco-Latin squares. *Sankhya*, 1938, vol. 3. no. 4, pp. 323–338.
15. Bose R.C., Shrikhande S.S., Parker E.T. Further results on the construction of mutually orthogonal Latin squares and the falsity of Euler's conjecture. *Canadian Journal of Mathematics*, 1960, pp. 189-203.
16. Colbourn C.J., Dinitz J.H. *Handbook of combinatorial designs*. CRC Press. 2010. http://doi.org/10.1201/9781420010541

17. Cook S.A. The complexity of theorem-proving procedures. *Third annual ACM symposium on Theory of computing*, 1971, Ohio, USA. pp. 151-159.
18. DIMACS Implementation Challenges [Electronic resource]. URL: http://dimacs.rutgers.edu/Challenges/
19. Egan J., Wanless I. M. Enumeration of MOLS of small order. *Math. Comput*, 2016, vol. 85, pp. 799–824. https://doi.org/10.1090/mcom/3010
20. Haken A. The intractability or resolution. *Theoretical Computer Science*, 1985, vol. 39. pp. 297–308.
21. Heule M.J.H., Kullman O., Marec V.W. Solving and Verifying the Boolean Pythagorean Triples Problem via Cube-and-Conquer. *Lecture Notes in Computer Science*, 2014, vol. 9710 (SAT-2016), pp. 228-245.
22. Konev B., Lisitsa A. A SAT Attack on Erdos Discrepancy Problem. *Lecture Notes in Computer Science*, 2014, vol. 8561 (SAT-2014), pp. 219-226. https://doi.org/10.1007/978-3-319-09284-3_17
23. Lam C.W.H. The Search for a Finite Projective Plane of Order 10. *The American Mathematical Monthly*. 1991. pp. 305-318.
24. Marques-Silva J. P., Lynce I., Malik S. Conflict-Driven Clause Learning SAT solvers. *Handbook of Satisfiability*, 2009, pp. 131–153. https://doi.org/10.3233/978-1-58603-929-5-131
25. McKay B. D., Wanless I. M.A census of small Latin hypercubes. *SIAM J. Discrete Math.*, 2008, vol. 22, pp. 719–736. https://doi.org/10.1137/070693874
26. Mironov I., Zhang L. Applications of SAT solvers to cryptanalysis of hash functions. *Lecture Notes in Computer Science*, 2006, vol. 4121, pp. 102-115. https://doi.org/10.1007/11814948_13
27. Semenov A., Zaikin O., Otpuschennikov I., Kochemazov S., Ignatiev A. On cryptographic attacks using backdoors for SAT. *In The Thirty-Second AAAI Conference on Artificial Intelligence, AAAI'2018*, 2018, pp. 6641-6648.
28. Vatutin E., Kochemazov S., Zaikin O., Valyaev S. Enumerating the Transversals for Diagonal Latin Squares of Small Order. *Ceur-WS*, 2017, vol. 1973, pp. 6-14.
29. Vatutin E., Kochemazov S., Zaikin O. Applying Volunteer and Parallel Computing for Enumerating Diagonal Latin Squares of Order 9. *Communication in Computer and Information Science*, 2017, vol. 753, pp. 114-129. https://doi.org/10.1007/978-3-319-67035-5_9
30. Williams R., Gomes C.P., Selman B. Backdoors to typical case complexity. *Proceedings of the 18th International Joint Conference on Articial Intelligence - IJCAI 2003*, 2003, pp. 1173-1178.

Evgenia Beley, Undergraduate, Irkutsk National Research Technical University, 83, Lermontov st., Irkutsk, 664074, Russian Federation, tel.: (908)6684039 (e-mail: `zhenyha28@mail.ru`)

Alexander Semenov, Candidate of Sciences (Technic Sciences), Head of Laboratory, Matrosov Institute for System Dynamics and Control Theory SB RAS, 134, Lermontov st., Irkutsk, 664033, Russian Federation, tel.: (914)8707647 (e-mail: `biclop.rambler@yandex.ru`)

Received 30.04.19