МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПРИКЛАДНАЯ ДИСКРЕТНАЯ МАТЕМАТИКА

Научный журнал

2025 № 70

Зарегистрирован в Федеральной службе по надзору
в сфере связи и массовых коммуникаций

Свидетельство о регистрации ПИ № ФС 77-33762 от 16 октября 2008 г.

Подписной индекс в объединённом каталоге «Пресса России» 38696

УЧРЕДИТЕЛЬ
Томский государственный университет

Адрес редакции и издателя: 634050, г. Томск, пр. Ленина, 36
E-mail: pank@mail.tsu.ru

В журнале публикуются результаты фундаментальных и прикладных научных исследований отечественных и зарубежных ученых, включая студентов и аспирантов, в области дискретной математики и её приложений в криптографии, компьютерной безопасности, кибернетике, информатике, программировании, теории надёжности, интеллектуальных системах.

Периодичность выхода журнала: 4 номера в год.

Редактор *Н. И. Шидловская*
Редактор-переводчик *Т. В. Бутузова*
Верстка *И. А. Панкратовой*

Подписано к печати 28.11.2025. Формат $60 \times 84\frac{1}{8}$. Усл. п. л. 14,8. Тираж 300 экз.
Заказ № 6578. Цена свободная. Дата выхода в свет 10.12.2025.

Отпечатано на оборудовании
Издательства Томского государственного университета
634050, г. Томск, пр. Ленина, 36
Тел.: 8(3822)53-15-28, 52-98-49

СОДЕРЖАНИЕ

CONTENTS

ПРИКЛАДНАЯ ТЕОРИЯ КОДИРОВАНИЯ

УДК 510.52 DOI 10.17223/20710410/70/1

О ПРОБЛЕМЕ ДЕКОДИРОВАНИЯ ПО ПРИНЦИПУ МАКСИМАЛЬНОГО ПРАВДОПОДОБИЯ

В. Ю. Попов

Уральский федеральный университет имени первого Президента России Б. Н. Ельцина, г. Екатеринбург, Россия

E-mail: popovvvv@gmail.com

Рассмотрен количественный аналог проблемы декодирования по принципу максимального правдоподобия. Установлена экономная сводимость от проблемы совершенного паросочетания и слабо экономная сводимость от проблемы максимального разреза. Показана полнота в классах **WPP**, $\mathbf{C}_{=}\mathbf{P}$ и **PP** для некоторых количественных вариантов проблемы декодирования по принципу максимального правдоподобия.

Ключевые слова: *вычислительная сложность, проблема декодирования по принципу максимального правдоподобия, проблема совершенного паросочетания, проблема максимального разреза.*

ON THE MAXIMUM-LIKELIHOOD DECODING PROBLEM

V. Yu. Popov

Ural Federal University named after the first President of Russia B. N. Yeltsin, Ekaterinburg, Russia

Maximum likelihood decoding is an extensively used technique for digital communication systems. The hardness of the maximum likelihood decoding problem is the fundamental basis of the security justification for code-based cryptography. The study of code-based cryptography algorithms is considered as one of the main directions in the development of post-quantum cryptography. Nevertheless, it is known relatively little about the hardness of the maximum likelihood decoding problem. In this paper, we present an alternative proof of the **NP**-completeness of the maximum likelihood decoding problem that provides additional evidence for the security of cryptographic algorithms based on Classic McEliece. Also, we consider the counting variant of the maximum likelihood decoding problem, an important tool for finding collisions. We obtain a parsimonious reduction from the perfect matching problem and a weakly parsimonious reduction from the simple Max Cut problem. As a consequence, we obtain the #**P**-completeness of the counting variant of the maximum likelihood decoding problem. Also, we consider some counting variants of the maximum likelihood decoding problem for classes of computational complexity that are of interest from the point of view of quantum computing and post-quantum cryptography. In particular, we obtain completeness results for classes **WPP**, $\mathbf{C}_{=}\mathbf{P}$, and **PP**.

Keywords: *computational complexity, maximum likelihood decoding problem, perfect matching problem, Max Cut problem.*

Введение

Проблема декодирования по принципу максимального правдоподобия является одной из ключевых алгоритмических проблем теории кодирования. От качества её решения зависит эффективность применения кодов. Однако, как показано в работе [1], эта проблема является вычислительно трудной даже для двоичных линейных кодов. С другой стороны, трудность этой проблемы стала обоснованием криптографической стойкости алгоритма шифрования, предложенного в [2]. Схема алгоритма [2] оказалась весьма популярной и продуктивной. Она получила существенное дальнейшее развитие (см., например, [3–5]). В частности, в конкурсе NIST (National Institute of Standards and Technology) [6] в качестве кандидатов на постквантовый криптографический стандарт участвовали алгоритмы Classic McEliece и NTS-KEM [7, 8]. Алгоритм Classic McEliece, по сути дела, отражает изначально предложенную в работе [2] идею. Алгоритм NTS-KEM является существенной переработкой алгоритмов [2, 9] на основе преобразования, подобного преобразованиям [10, 11]. Однако его криптографическая стойкость во многом опирается на тот же фундамент, что и у Classic McEliece. Существенное внимание исследователи проявляют и к разработке для алгоритма [2] квантовых аналогов (см., например, [12, 13]).

Развитие квантовых технологий в последние годы значительно усилило интерес к проблеме декодирования с различных точек зрения. Существующие и перспективные технологии квантовых вычислений критически зависят от методов кодирования и эффективности исправления ошибок. Этим обусловлены интенсивное развитие квантовой теории кодирования и разработка новых квантовых кодов [14–16]. Возможная стойкость Classic McEliece относительно квантовых атак обусловила ряд исследований по криптоанализу [17–19]. С другой стороны, для нахождения более надёжного алгоритма существенное внимание уделяется адаптации Classic McEliece к новым кодам [20–22]. Исходя из этого, возросла важность дополнительного исследования вычислительной сложности проблемы декодирования как в классическом, так и в квантовом случае. Для квантовой проблемы декодирования рассмотрен ряд формулировок, для которых получены результаты не только по **NP**-трудности [23–25], но и по #**P**-трудности [26, 27]. Значительный интерес проявляется к поиску эффективных методов решения проблемы декодирования. Рассматриваются различные подходы к решению проблемы декодирования не только для классических кодов, но и для квантовых [28–30]. Среди используемых методов можно выделить обучение с подкреплением [31, 32], глубокое обучение [33] и различные типы нейронных сетей [34, 35]. В частности, можно отметить активное применение свёрточных сетей [36, 37]. Следует отметить, что гипотеза о трудности решения проблемы декодирования, так же как и проблемы обучения с ошибками (LWE), используемой для обоснования надёжности криптографических алгоритмов на решётках, является некоторым противопоставлением гипотезе об эффективности машинного обучения [38]. Многочисленные попытки применения методов машинного обучения для решения задач, связанных с декодированием, в значительной мере обусловлены уверенностью в недостаточной обоснованности трудности проблемы декодирования с практической точки зрения, что неоднократно отмечалось исследователями [39–41].

На практике как в теории кодирования, так и в криптографии обычно предполагается, что ошибка должна быть сравнительно мала. Желательно, чтобы она была

меньше минимального расстояния кода. Для исправления w ошибок необходимо минимальное расстояние $2w+1$. Для проблемы декодирования **NP**-трудность возможности декодирования доказана лишь для очень больших значений ошибки, которые наверняка не встретятся на практике.

Вопрос о количестве вариантов декодирования является принципиально важным для практического применения. От его решения зависит качество декодирования получателем информации. Кроме того, задача выяснения количества вариантов декодирования возникает при поиске возможных коллизий, что делает её важной с криптографической точки зрения. Однако количественные варианты проблемы декодирования даже не рассматривались. В частности, в отличие от квантовых кодов, #**P**-трудность для классических двоичных кодов не доказана. Следует отметить, что трудность количественных версий не следует автоматически из **NP**-трудности проблемы. Интересным примером является проблема NAESAT [42, п. 9.2]. Эта проблема является **NP**-полной [42, теорема 9.3]. Вариант NAESAT, требующий выяснить единственность решения, разрешим за полиномиальное время [43], а вариант NAESAT, требующий найти количество решений, является #**P**-полной проблемой [44].

В данной работе доказана вычислительная трудность вопроса о количестве вариантов декодирования.

1. Основные определения

Двухэлементное поле будем обозначать через $\mathbb{Z}_2$; вес Хэмминга $\mathrm{wt}(x)$ вектора $x \in \mathbb{Z}_2^n$ — количество его ненулевых координат. Рассмотрим формальное определение проблемы декодирования по принципу максимального правдоподобия.

Maximum Likelihood Decoding (MLD)

Дано: Двоичная $(m \times n)$-матрица $H \in \mathbb{Z}_2^{m \times n}$, вектор $s \in \mathbb{Z}_2^m$, число $k \in \mathbb{N}$.

Вопрос: Существует ли вектор $x \in \mathbb{Z}_2^n$, такой, что $Hx = s$ и $\mathrm{wt}(x) \leqslant k$?

Следуя [45], подсчитывающей машиной Тьюринга будем называть стандартную недетерминированную машину Тьюринга с дополнительной лентой, предназначенной для печати в двоичном виде количества допустимых вычислений этой машины для данного входа. Пусть максимальное время допустимого вычисления на входах, размер которых не превосходит n, равно $t(n)$. Предполагается, что подсчитывающая машина Тьюринга в худшем случае имеет сложность по времени $t(n)$. Таким образом, трудоёмкость генерации количества допустимых вычислений на дополнительной ленте не учитывается.

Класс #**P** состоит из всех функций, которые могут быть вычислены подсчитывающей машиной Тьюринга за полиномиальное время [45]. Количественная версия проблемы MLD может быть сформулирована следующим образом:

#MLD

Дано: Двоичная $(m \times n)$-матрица $H \in \mathbb{Z}_2^{m \times n}$, вектор $s \in \mathbb{Z}_2^m$, число $k \in \mathbb{N}$.

Найти: Количество попарно различных векторов $x \in \mathbb{Z}_2^n$, таких, что $Hx = s$ и $\mathrm{wt}(x) \leqslant k$.

Стандартный подход к доказательству #**P**-трудности некоторой алгоритмической проблемы A заключается в сведении к ней некоторой #**P**-полной проблемы B. Однако в отличие от **NP**-трудности, кроме полиномиальности сводимости должно выполняться требование сохранения количества решений [42]. В некоторых случаях необходимо более точное определение условия, накладываемого на полиномиальную сводимость. Для произвольных исходных данных I алгоритмической проблемы Π обозначим че-

рез $\#I$ количество решений проблемы Π на входе I. Следуя [46], полиномиальную сводимость R проблемы Π_1 к проблеме Π_2 будем называть слабо экономной, если существует полиномиально вычислимая функция f_R, такая, что для любых входов $I_1 \in \Pi_1$ и $I_2 \in \Pi_2$, таких, что $R(I_1) = I_2$, имеет место равенство $\#I_1 = f_R(I_1)\#I_2$. Полиномиальная сводимость R проблемы Π_1 к проблеме Π_2 называется экономной, если для любых входов $I_1 \in \Pi_1$ и $I_2 \in \Pi_2$, таких, что $R(I_1) = I_2$, имеет место равенство $\#I_1 = \#I_2$.

2. Анализ полиномиальных сводимостей для проблемы декодирования

Многие известные полиномиальные сведения сохраняют количество решений. В этих случаях доказательство **NP**-трудности является и доказательством #**P**-трудности [42]. Поэтому имеет смысл проанализировать известные полиномиальные сводимости для проблемы MLD.

Доказательство **NP**-трудности проблемы MLD, предложенное в работе [1], основано на полиномиальном сведении проблемы существования трёхмерного сочетания, **NP**-трудность которой доказана в [47]. Проблема существования трёхмерного сочетания может быть сформулирована следующим образом.

3-Dimensional Matching (3DM)

Дано: Натуральное число $n \in \mathbb{N}$, множество T, такое, что $|T| = n$, семейство трёхэлементных множеств $U \subseteq T \times T \times T$.

Вопрос: Существует ли $W \subseteq U$, такое, что $|W| = n$ и никакие два элемента W не имеют общих компонент?

Предположим, что $T = \{1, 2, \ldots, n\}$, $U = \{S_1, S_2, \ldots, S_m\}$. Полиномиальное сведение проблемы 3DM к MLD определяет матрица H размера $m \times 3n$, такая, что $S_i = (a, b, c)$ тогда и только тогда, когда в i-й строке матрицы H ровно три единицы и эти единицы располагаются в столбцах с номерами a, $b + n$, $c + 2n$. Следуя рассуждениям [1], легко проверить, что сумма менее чем n строк не даёт вектор, состоящий из одних единиц, а сумма n строк, дающая вектор, состоящий из одних единиц, находится во взаимно однозначном соответствии с некоторым решением проблемы 3DM. Таким образом, полиномиальное сведение проблемы 3DM к проблеме MLD сохраняет количество решений. Рассмотрим теперь полиномиальные сведения к проблеме 3DM.

В работе [47] доказательство **NP**-трудности проблемы 3DM проведено последовательным полиномиальным сведением от проблемы выполнимости через проблемы 3-выполнимости, хроматического числа и точного покрытия. Появление проблемы хроматического числа в последовательности полиномиальных сведений очевидным образом делает эту последовательность сведением, не сохраняющим количество решений, поскольку вместе с каждой допустимой раскраской решением будет и любая взаимно однозначная перестановка цветов. Однако гипотетически это можно обойти аналогично тому, как это сделано для проблемы вычисления перманента [42, теорема 18.3], устанавливая соответствие между количеством решений t проблемы 3-выполнимости и количеством решений $c!d$ проблемы хроматического числа, где c обозначает количество цветов. Поэтому мы остановимся на полиномиальном сведении 3-выполнимости к проблеме хроматического числа более подробно. Рассмотрим проблемы 3-выполнимости и хроматического числа в формулировках, соответствующих работе [47].

SATisfiability with at most 3 literals per clause ($\leqslant$3SAT)

Дано: Булева функция $f(x_1, x_2, \ldots, x_m) = \bigwedge\limits_{i=1}^{r} D_i$, где для любого $1 \leqslant i \leqslant r$ функция D_i является дизъюнкцией не более трёх литералов из множества переменных и их отрицаний $\{x_1, x_2, \ldots, x_m, \neg x_1, \neg x_2, \ldots, \neg x_m\}$.

Вопрос: Существует ли набор значений переменных $x_j \in \{0, 1\}$, $1 \leqslant j \leqslant m$, такой, что выполняется равенство $f(x_1, x_2, \ldots, x_m) = 1$?

Chromatic Number (CN)

Дано: Граф $G = (V, E)$, заданный множеством вершин V и множеством рёбер E, натуральное число $k \in \mathbb{N}$.

Вопрос: Существует ли функция $\phi : \mathbb{N} \to \{1, 2, \ldots, k\}$, такая, что $\phi(u) \neq \phi(v)$ для любого ребра $(u, v) \in E$?

В предположении $m \geqslant 4$ в работе [47] полиномиальное сведение проблемы $\leqslant$3SAT к проблеме CN задаётся следующими соотношениями:

$$V = \{x_1, x_2, \ldots, x_m, \} \cup \{\neg x_1, \neg x_2, \ldots, \neg x_m\} \cup \{v_1, v_2, \ldots, v_m\} \cup \{D_1, D_2, \ldots, D_r\},$$
$$E = \{(x_i, \neg x_i) : 1 \leqslant i \leqslant m\} \cup$$
$$\cup \{(v_i, v_j) : 1 \leqslant i \leqslant m,\ 1 \leqslant j \leqslant m,\ i \neq j\} \cup$$
$$\cup \{(v_i, x_j) : 1 \leqslant i \leqslant m,\ 1 \leqslant j \leqslant m,\ i \neq j\} \cup$$
$$\cup \{(v_i, \neg x_j) : 1 \leqslant i \leqslant m,\ 1 \leqslant j \leqslant m, i \neq j\} \cup$$
$$\cup \{(x_i, D_j) : 1 \leqslant i \leqslant m,\ 1 \leqslant j \leqslant r,\ x_i \notin D_j\} \cup$$
$$\cup \{(\neg x_i, D_j) : 1 \leqslant i \leqslant m,\ 1 \leqslant j \leqslant r,\ \neg x_i \notin D_j\},$$
$$k = m + 1.$$

Включение $\{(v_i, v_j) : 1 \leqslant i \leqslant m,\ 1 \leqslant j \leqslant m,\ i \neq j\} \subset E$ гарантирует, что $\phi(v_i) \neq \phi(v_j)$ для всех $1 \leqslant i \leqslant m$, $1 \leqslant j \leqslant m$, $i \neq j$. Так как значения функции ϕ для всех вершин из множества $\{v_1, v_2, \ldots, v_m\}$ попарно различны и $k = m+1$, имеется лишь одно значение функции ϕ, такое, что оно не является элементом множества $\{\phi(v_1), \phi(v_2), \ldots, \phi(v_m)\}$. Обозначим это значение через a. Включение

$$\{(v_i, x_j) : 1 \leqslant i \leqslant m,\ 1 \leqslant j \leqslant m,\ i \neq j\} \cup \{(v_i, \neg x_j) : 1 \leqslant i \leqslant m,\ 1 \leqslant j \leqslant m,\ i \neq j\} \subset E$$

гарантирует, что для всех $1 \leqslant i \leqslant m$, $1 \leqslant j \leqslant m$, $i \neq j$ выполняется $\phi(v_i) \neq \phi(x_j)$ и $\phi(v_i) \neq \phi(\neg x_j)$. Поэтому $\phi(x_j) \in \{\phi(v_j), a\}$ и $\phi(\neg x_j) \in \{\phi(v_j), a\}$ для всех $1 \leqslant j \leqslant m$.

Из включения $\{(x_i, \neg x_i) : 1 \leqslant i \leqslant m\} \subset E$ получаем $\{\phi(x_j), \phi(\neg x_j)\} = \{\phi(v_j), a\}$ для всех $1 \leqslant j \leqslant m$. Поскольку $m \geqslant 4$ и дизъюнкции содержат не более трёх литералов, для любого i найдётся такое j, что $x_j \notin D_i$ и $\neg x_j \notin D_i$. Тогда из условия

$$\{(x_i, D_j) : 1 \leqslant i \leqslant m, 1 \leqslant j \leqslant r, x_i \notin D_j\} \cup \{(\neg x_i, D_j) : 1 \leqslant i \leqslant m, 1 \leqslant j \leqslant r, \neg x_i \notin D_j\} \subset E$$

получим, что $\phi(D_i) \neq a$ для всех $1 \leqslant i \leqslant r$. Кроме того, отсюда же следует, что

$$\phi(D_i) \in \{\phi(v_j) : x_j \in D_i,\ 1 \leqslant j \leqslant m\} \cup \{\phi(v_j) : \neg x_j \in D_i,\ 1 \leqslant j \leqslant m\}$$

для всех $1 \leqslant i \leqslant r$. Таким образом, функция $\phi : \mathbb{N} \to \{1, 2, \ldots, k\}$, такая, что $\phi(u) \neq \phi(v)$ для любого ребра $(u, v) \in E$, существует тогда и только тогда, когда для любой дизъюнкции найдётся литерал w с условием $\phi(w) \neq a$. Мы можем интерпретировать a как 0, рассматривая все элементы множества $\{\phi(v_1), \phi(v_2), \ldots, \phi(v_m)\}$ как 1.

Это позволяет убедиться в том, что данное полиномиальное сведение не сохраняет количество решений. В частности, значение $\phi(D_i)$ может совпадать со значением ϕ для любого истинного литерала из дизъюнкции D_i.

Кроме доказательства **NP**-трудности проблемы 3DM, предложенного в работе [47], имеется доказательство, приведённое в [42, 48], через полиномиальное сведение стандартной версии проблемы 3-выполнимости, предполагающей наличие ровно трёх литералов в каждой дизъюнкции. Однако это полиномиальное сведение тоже не сохраняет количество решений. В частности, в обозначениях [48, теорема 3.2] каждой дизъюнкции c_j соответствует множество

$$C_j = \{(u_i[j], s_1[j], s_2[j]) : u_i \in c_j\} \cup \{(\overline{u}_i[j], s_1[j], s_2[j]) : (\overline{u}_i \in c_j\}.$$

Из этого множества в трёхмерное сочетание M' можно выбрать любой элемент, соответствующий истинному литералу, что позволяет по одному решению проблемы 3-выполнимости построить несколько различных трёхмерных сочетаний.

Мы убедились в том, что имеющиеся для проблемы 3DM сводимости не подходят для доказательства #**P**-трудности проблемы #MLD. Однако сводимость, построенную в работе [1], несложно адаптировать для проблемы совершенного паросочетания для двудольного графа. Рассмотрим несколько более общий подход, установив сводимость для произвольного графа.

Perfect Matching (PM)

Дано: Граф $G = (V, E)$, заданный множеством вершин V и множеством рёбер E.

Вопрос: Существует ли $M \subseteq E$, такое, что каждая вершина графа G инцидентна ровно одному ребру из множества M?

Теорема 1. Существует экономная сводимость проблемы PM к проблеме MLD.

Доказательство. Без ограничения общности можем полагать, что граф имеет чётное количество вершин. Рассмотрим граф $G = (V, E)$, заданный множеством вершин $V = \{v_1, v_2, \ldots, v_n\}$ и множеством рёбер $E = \{e_1, e_2, \ldots, e_m\}$. Для графа G рассмотрим двоичную матрицу H размера $m \times n$, такую, что её элемент $h_{i,j}$ равен единице тогда и только тогда, когда вершина v_j инцидентна ребру e_i. Пусть $k = n/2$. Будем полагать, что вектор s состоит из одних единиц. Легко понять, что вектор $x \in \mathbb{Z}_2^n$, такой, что $Hx = s$ и $\mathrm{wt}(x) \leqslant k$, существует тогда и только тогда, когда в графе G существует совершенное паросочетание, т. е. такое подмножество $M \subseteq E$, что каждая вершина графа G инцидентна ровно одному ребру из множества M. Более того, рёбра множества M и единицы вектора x находятся во взаимно однозначном соответствии. ■

В работе [45] доказано, что проблема вычисления перманента двоичной матрицы является #**P**-полной относительно полиномиальной сводимости. Поскольку значение перманента двоичной матрицы равно количеству совершенных паросочетаний в двудольном графе, заданном этой матрицей, из теоремы 1 и результата [45] вытекает

Следствие 1. Проблема #MLD является #**P**-полной относительно полиномиальной сводимости.

Кроме полиномиального сведения, предложенного в [1], существует ещё один подход к обоснованию **NP**-трудности проблемы MLD. Он основан на использовании **NP**-трудности проблемы существования максимального разреза, которая может быть сформулирована следующим образом [39, 42].

Simple Max Cut (SMC)

Дано: Натуральное число $n \in \mathbb{N}$, граф $G = (V, E)$.

Вопрос: Существует ли такое множество вершин $W \subseteq V$, что $|F| \geqslant n$, где $F = \{(u, v) : u \in W,\ v \in V \setminus W\}$?

NP-полнота проблемы SMC доказана в работе [49] (см. также [42, теорема 9.5]). В [50] предложен подход к построению кодов по графам. В [39] указано, что проблема MLD является **NP**-полной, и это можно доказать, используя матрицу смежности графа и вектор, состоящий из одних единиц [39, предложение 1]. Идея [39] получила поддержку исследователей (см., например, [51]). Кроме того, на основе **NP**-полноты проблемы SMC предложен ряд подходов к получению альтернативных доказательств **NP**-трудности проблемы MLD через промежуточные сведения [52, 53]. Однако подход, предложенный в [39], без некоторой доработки использовать нельзя. Например, можно рассмотреть полный трёхвершинный граф K_3. Очевидно, что для K_3 максимальный разрез равен 2. В то же время граф K_3 можно представить в виде

$$K_3 = (V, E), \quad V = \{a_1, a_2, a_3\}, \quad E = \{b_1, b_2, b_3\},$$
$$b_1 = (a_1, a_2), \quad b_2 = (a_1, a_3), \quad b_3 = (a_2, a_3).$$

В этом случае строки матрицы H имеют вид $(1, 1, 0)$, $(1, 0, 1)$, $(0, 1, 1)$. Матричное уравнение $Hx = s$ для вектора s, состоящего из одних единиц, равносильно системе уравнений

$$\begin{cases} x_1 + x_2 = 1, \\ x_1 + x_3 = 1, \\ x_2 + x_3 = 1, \end{cases}$$

которая, очевидно, не имеет решений. Аналогичная ситуация имеет место для любого графа, не являющегося двудольным, а для двудольных графов максимальный разрез всегда равен количеству рёбер графа, что тривиально влечёт разрешимость проблемы SMC для двудольных графов за линейное время.

Непосредственное применение подхода из [39] позволяет получить альтернативное доказательство **NP**-трудности известной проблемы существования ближайшего кодового слова, которую можно сформулировать следующим образом [54, 55].

Nearest codeword (NC)

Дано: Двоичная матрица $H \in \mathbb{Z}_2^{m \times n}$, вектор $s \in \mathbb{Z}_2^m$, число $k \in \mathbb{N}$.

Вопрос: Существует ли вектор такой $x \in \mathbb{Z}_2^n$, что $\mathrm{wt}(Hx + s) \leqslant k$?

3. Полиномиальная сводимость для проблемы максимального разреза

Проблема MLD активно используется в криптографических целях. В частности, большое количество криптографических алгоритмов построено на основе схемы [2]. Однако полиномиальная сводимость, установленная в [1], гарантирует трудность проблемы MLD лишь для исходных данных, которые не могут быть использованы на практике. Поэтому представляет значительный практический интерес нахождение альтернативных доказательств **NP**-трудности проблемы MLD. Учитывая существенное внимание к использованию проблемы SMC для обоснования трудности проблемы MLD, установим корректность полиномиальной сводимости проблемы SMC к проблеме MLD.

Для произвольного алфавита Σ обозначим через Σ^n, где $n \in \mathbb{N}$, множество всевозможных слов длины n в алфавите Σ. Обозначим через 0^n и 1^n, где $n \in \mathbb{N}$, единственные элементы множеств $\{0\}^n$ и $\{1\}^n$ соответственно.

Рассмотрим граф $G = (V, E)$ без петель и кратных рёбер, заданный множеством вершин $V = \{v_1, v_2, \ldots, v_p\}$ и множеством рёбер $E = \{e_1, e_2, \ldots, e_q\}$. Каждой вершине v_i поставим в соответствие слово $h_i \in \{0,1\}^{(p+1)q}$. Пусть для любого $1 \leqslant i \leqslant p$ слово h_i имеет вид $h_i = a_{i,1}a_{i,2}\ldots a_{i,q}$, где $a_{i,j} = 1^{p+1}$, если $e_j = (v_i, v_k)$ для некоторого $k \in \in \{1, \ldots, p\}$, и $a_{i,j} = 0^{p+1}$, если $e_j \neq (v_i, v_k)$ для любого $k \in \{1, \ldots, p\}$. Определим слово

$$h_i = h_{i,1}h_{i,2}\ldots h_{i,(p+1)q} \in \{0,1\}^{(p+1)q}$$

для любого $p+1 \leqslant i \leqslant p+(p+1)q$, полагая

$$h_{i,j} = \begin{cases} 1, & j = i-p,\ 1 \leqslant j \leqslant (p+1)q, \\ 0, & j \neq i-p,\ 1 \leqslant j \leqslant (p+1)q. \end{cases} \tag{1}$$

Пусть вектор s состоит из одних единиц, $m = p+(p+1)q$, $n = (p+1)q$. С помощью слов h_i, $1 \leqslant i \leqslant m$, определим матрицу H размера $m \times n$, где i-я строка матрицы H равна слову h_i для любого $1 \leqslant i \leqslant m$. Заметим, что подматрица размера $n \times n$ матрицы H, состоящая из строк h_l, где $p+1 \leqslant l \leqslant m$, в силу соотношения (1) является единичной.

Покажем, что максимальный разрез графа G не меньше r тогда и только тогда, когда существует такой вектор $x \in \mathbb{Z}_2^n$, что $Hx = s$ и $\mathrm{wt}(x) \leqslant k$, где $k = p+(q-r)(p+1)$.

Допустим, что максимальный разрез графа G равен r. Без ограничения общности можно считать, что $r > 0$. Тогда существует разбиение $V = V_1 \cup V_2$ на непустые подмножества V_1 и V_2, такие, что

$$|\{(u,v) : u \in V_1,\ v \in V_2,\ (u,v) \in E\}| \geqslant r. \tag{2}$$

Каждому ребру $e_t = (v_i, v_j) \in E$, $1 \leqslant t \leqslant q$, поставим в соответствие набор столбцов матрицы H с номерами от $(t-1)(p+1)+1$ до $(t-1)(p+1)+p+1$. В каждом из этих столбцов ровно три единицы: две единицы в первых p строках столбца (по одной на каждую вершину ребра e_t), что следует из определения слов h_l при $1 \leqslant l \leqslant p$, и одна единица в последних n строках в силу единичности подматрицы, состоящей из строк h_l, где $p+1 \leqslant l \leqslant m$. Эти единицы расположены в строках с номерами i, j, $l+p$, где l — номер столбца. Поэтому в i-й строке, соответствующей вершине v_i, для каждого инцидентного вершине $v_i \in V_1$ ребра e_t имеется группа из $p+1$ единиц в таких столбцах, что все остальные строки, соответствующие вершинам из V_1, имеют в этих столбцах нули. Отсюда и из неравенства (2) получаем соотношение

$$\mathrm{wt}\left(\sum_{v_i \in V_1} h_i\right) \geqslant r(p+1). \tag{3}$$

Из неравенства (3) очевидным образом следует, что вектор

$$\sum_{v_i \in V_1} h_i \tag{4}$$

имеет не более $n-r(p+1)$ нулевых координат. Обозначим через N множество нулевых координат вектора (4). По определению вектора s из (1) получаем равенство

$$\sum_{v_i \in V_1} h_i + \sum_{j-p \in N} h_j = s. \tag{5}$$

Рассмотрим вектор

$$x = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}, \tag{6}$$

такой, что для любого i, $1 \leqslant i \leqslant n$, равенство $x_i = 1$ выполняется тогда и только тогда, когда $v_i \in V_1$ или $i - p \in N$. Заметим, что соотношение $i - p \in N$ гипотетически может выполняться лишь при $i \geqslant p + 1$. По определению матрицы H получаем равенство

$$Hx = \sum_{v_i \in V_1} h_i + \sum_{j-p \in N} h_j. \tag{7}$$

Из (5) и (7) получаем равенство $Hx = s$. По определению вектора x имеет место

$$\mathrm{wt}(x) = |V_1| + |N|. \tag{8}$$

Поскольку вектор (4) имеет не более $n - r(p+1)$ нулевых координат, по определению множества N должно выполняться неравенство

$$|N| \leqslant n - r(p+1). \tag{9}$$

Мощность множества V_1 не превосходит мощности множества всех вершин графа G. Поэтому из соотношений (8) и (9) следует неравенство $\mathrm{wt}(x) \leqslant p + n - r(p+1)$. Так как $n = (p+1)q$, $\mathrm{wt}(x) \leqslant p + (p+1)q - r(p+1) = p + (q-r)(p+1)$, что и требовалось.

Предположим теперь, что существует вектор (6), такой, что $Hx = s$ и

$$\mathrm{wt}(x) \leqslant p + (q-r)(p+1). \tag{10}$$

Пусть

$$P = \{i : x_i = 1,\ i \leqslant p\}, \quad Q = \{i : x_i = 1,\ i > p\}. \tag{11}$$

Из (11) по определению матрицы H получаем равенство

$$Hx = \sum_{i \in P} h_i + \sum_{j \in Q} h_j.$$

По предположению $Hx = s$, поэтому $s = \sum\limits_{i \in P} h_i + \sum\limits_{j \in Q} h_j$. Отсюда по определению вектора s получаем соотношение

$$\mathrm{wt}\left(\sum_{i \in P} h_i + \sum_{j \in Q} h_j\right) = (p+1)q. \tag{12}$$

По определению расстояния Хэмминга выполняется неравенство

$$\mathrm{wt}\left(\sum_{i \in P} h_i + \sum_{j \in Q} h_j\right) \leqslant \mathrm{wt}\left(\sum_{i \in P} h_i\right) + \mathrm{wt}\left(\sum_{j \in Q} h_j\right). \tag{13}$$

По определению матрицы H из (11) получаем

$$\mathrm{wt}\left(\sum_{j \in Q} h_j\right) = |Q|. \tag{14}$$

Из соотношений (6), (11) и (14) следует, что $\mathrm{wt}\Big(\sum\limits_{j \in Q} h_j\Big) \leqslant \mathrm{wt}(x)$. Отсюда по предположению (10) получаем

$$\mathrm{wt}\left(\sum_{j \in Q} h_j\right) \leqslant p + (q-r)(p+1). \tag{15}$$

Из соотношений (12) и (13) очевидным образом следует, что

$$(p+1)q \leqslant \mathrm{wt}\left(\sum_{i\in P} h_i\right) + \mathrm{wt}\left(\sum_{j\in Q} h_j\right). \tag{16}$$

Из (15) и (16) можно получить неравенство

$$(p+1)q \leqslant \mathrm{wt}\left(\sum_{i\in P} h_i\right) + p + (q-r)(p+1). \tag{17}$$

Соотношение (17) равносильно неравенству

$$(r-1)(p+1) + 1 \leqslant \mathrm{wt}\left(\sum_{i\in P} h_i\right). \tag{18}$$

Множество P задает множество вершин $V(P) = \{v_i : i \in P\}$, которое определяет некоторый разрез $E' \subseteq E$ графа G. Пусть $\sum\limits_{i\in P} h_i = (a_1, a_2, \ldots, a_{(p+1)q})$. Легко понять, что для произвольного t, $1 \leqslant t \leqslant q$, выполнимость равенства $a_j = 1$ для какого-либо j, удовлетворяющего условию

$$(t-1)(p+1) + 1 \leqslant j \leqslant (t-1)(p+1) + p + 1,$$

равносильна тому, что выполняется соотношение $e_t \in E'$ и равенство $a_i = 1$ имеет место для всех i, для которых справедливо

$$(t-1)(p+1) + 1 \leqslant i \leqslant (t-1)(p+1) + p + 1,$$

т. е. в векторе $(a_1, a_2, \ldots, a_{(p+1)q})$ все единичные координаты разбиваются на непересекающиеся группы, каждая из которых состоит из $p+1$ единиц и соответствует некоторому ребру e_t из множества E'. Следовательно, $\mathrm{wt}\left(\sum\limits_{i\in P} h_i\right) = (p+1)|E'|$. Поэтому из неравенства (18) следует требуемое соотношение $|E'| \geqslant r$.

Таким образом, мы установили полиномиальную сводимость проблемы SMC к проблеме MLD. Однако полученное сведение не сохраняет количество решений, поскольку для фиксированного разреза каждое разбиение $V = V_1 \cup V_2$ определяет два вектора x, а количество таких разбиений существенно зависит от свойств разреза. Более того, ясно, что никакая полиномиальная сводимость, развивающая предложенную в [39] идею кодирования векторов вершинами графа, не будет сохранять количество решений.

4. Количественная версия проблемы максимального разреза

Для количественного варианта проблемы SMC, пожалуй, наиболее естественно рассматривать требование нахождения количества разрезов. Однако нас интересует другая версия проблемы.

#SMC

Дано: Натуральное число $n \in \mathbb{N}$, граф $G = (V, E)$.

Найти: Количество множеств вершин $W \subseteq V$, таких, что выполняется неравенство $|F| \geqslant n$, где $F = \{(u, v) : (u, v) \in E, u \in W, v \in V \setminus W\}$.

Нетрудно убедиться, что полиномиальная сводимость проблемы NAESAT к проблеме SMC [42, теорема 9.5] является слабо экономной сводимостью проблемы

#NAESAT к проблеме #SMC. В частности, для любого решения проблемы #NAESAT мы получаем ровно два решения проблемы #SMC: W и $V \setminus W$. С учётом тривиальной принадлежности проблемы #SMC классу $\#\mathbf{P}$ отсюда и из $\#\mathbf{P}$-полноты проблемы #NAESAT относительно слабо экономной сводимости [44] вытекает $\#\mathbf{P}$-полнота проблемы #SMC относительно слабо экономной сводимости.

Легко проверить, что полиномиальная сводимость проблемы SMC к проблеме MLD, рассмотренная в п. 3, является экономной сводимостью проблемы #SMC к проблеме #MLD: каждому множеству вершин W, задающему разрез, соответствует единственный вектор x, и наоборот. Поскольку принадлежность проблемы #MLD классу $\#\mathbf{P}$ очевидна, отсюда и из $\#\mathbf{P}$-полноты проблемы #SMC вытекает

Теорема 2. Проблема #MLD является $\#\mathbf{P}$-полной относительно слабо экономной сводимости.

5. Полные проблемы для различных классов количественных проблем

Современная теория квантовых вычислений ассоциирует эффективную вычислимость с квантовой машиной Тьюринга и рассматривает класс **BQP** как класс эффективно решаемых задач [56, 57]. Точное соотношение класса **BQP** с классическими классами **P**, **NP** и **PSPACE** пока не выяснено. Однако господствует мнение, что $\mathbf{NP} \not\subseteq \mathbf{BQP}$. Это позволяет разрабатывать постквантовые криптографические алгоритмы, основываясь на трудности **NP**-полных проблем [57]. В то же время ведутся активные исследования в области специализированных квантовых вычислителей. В частности, архитектура D-Wave ориентирована на эффективное решение широкого класса **NP**-полных проблем [58–60]. Поэтому для важных криптографических моделей представляет значительный интерес исследование вычислительной сложности проблем за пределами класса **NP** [61, 62].

Для произвольной квантовой машины Тьюринга T обозначим через $P_{\text{acc}}(x)$ вероятность того, что на входе x машина T переходит в допустимое состояние. Соответственно через $P_{\text{rej}}(x)$ обозначим вероятность того, что машина T отвергает вход x. Следуя [63] (см. также [64, 65]), будем полагать, что

— **EQP** — класс языков $L \subseteq \Sigma^*$, для которых существует полиномиальная квантовая машина Тьюринга T, такая, что $x \in L \Rightarrow P_{\text{acc}}(x) = 1$ и $x \notin L \Rightarrow P_{\text{rej}}(x) = 1$ для любого $x \in \Sigma^*$;

— **BQP** — класс языков $L \subseteq \Sigma^*$, для которых существует полиномиальная квантовая машина Тьюринга T, такая, что $x \in L \Rightarrow P_{\text{acc}}(x) > 2/3$ и $x \notin L \Rightarrow P_{\text{rej}}(x) > 2/3$ для любого $x \in \Sigma^*$;

— **NQP** — класс языков $L \subseteq \Sigma^*$, для которых существует полиномиальная квантовая машина Тьюринга T, такая, что $x \in L \Rightarrow P_{\text{acc}}(x) > 0$ и $x \notin L \Rightarrow P_{\text{acc}}(x) = 0$ для любого $x \in \Sigma^*$.

Для классов **EQP**, **BQP** и **NQP** имеют место следующие очевидные включения:

$$\mathbf{EQP} \subseteq \mathbf{BQP} \subseteq \mathbf{NQP}.$$

Классы **EQP**, **BQP** и **NQP** обычно рассматриваются как квантовые аналоги классических классов **P**, **BPP** и **NP** соответственно [63]). Следует отметить, что алгоритмы Шора позволяют решать задачи факторизации и дискретного логарифма в классе **BQP** [66]. Обычно именно класс **BQP** рассматривается как класс алгоритмических проблем, которые могут быть эффективно решены на квантовой машине Тьюринга. Соответственно класс **NQP** следует рассматривать в рамках теоретически допустимой перспективы.

В работе [67] предложен модифицированный вариант квантовой машины Тьюринга, допускающий возможность измерения вероятности бита после завершения шага вычисления. Для этого варианта квантовой машины Тьюринга в [67] определён класс **PostBQP**, являющийся аналогом **BQP**. Хотя возможность практической реализации варианта машины, предложенного в [67], с точки зрения физики не считается реалистичной, сам класс **PostBQP** при определённых условиях может быть реализован на стандартных квантовых машинах. Поэтому класс **PostBQP**, как и класс **NQP**, следует учитывать в перспективных возможностях.

Определение класса **NP** и его криптографического подкласса **UP** через класс #**P** позволяет легко увидеть естественные аналоги **NP** и **UP**, расположенные выше в иерархии классов вычислительной сложности. Обозначим через **FP** класс разрешимых за полиномиальное время функциональных проблем. Для произвольной недетерминированной машины Тьюринга T обозначим через $\#\mathrm{acc}_T(x)$ и $\#\mathrm{rej}_T(x)$ количество допускающих и отвергающих вход x вычислений соответственно. Обозначим через **GapP** класс всех функций f, для которых существует недетерминированная машина Тьюринга T, такая, что $f(x) = \#\mathrm{acc}_T(x) - \#\mathrm{rej}_T(x)$ для всех x. Следуя [63, 68], дадим определение ряда классов вычислительной сложности как классов распознаваемых языков:

- **NP** — класс языков $L \subseteq \Sigma^*$, для которых существует функция $f \in \#\mathbf{P}$, такая, что $x \in L \Leftrightarrow f(x) > 0$ для любого $x \in \Sigma^*$;
- **UP** — класс языков $L \subseteq \Sigma^*$, для которых существует функция $f \in \#\mathbf{P}$, такая, что $x \in L \Rightarrow f(x) = 1$ и $x \notin L \Rightarrow f(x) = 0$ для любого $x \in \Sigma^*$;
- **PP** — класс языков $L \subseteq \Sigma^*$, для которых существует функция $f \in \mathbf{GapP}$, такая, что $x \in L \Leftrightarrow f(x) > 0$ для любого $x \in \Sigma^*$;
- **SPP** — класс языков $L \subseteq \Sigma^*$, для которых существует функция $f \in \mathbf{GapP}$, такая, что $x \in L \Rightarrow f(x) = 1$ и $x \notin L \Rightarrow f(x) = 0$ для любого $x \in \Sigma^*$;
- $\mathbf{C}_{=}\mathbf{P}$ — класс языков $L \subseteq \Sigma^*$, для которых существует функция $f \in \mathbf{GapP}$, такая, что $x \in L \Leftrightarrow f(x) = 0$ для любого $x \in \Sigma^*$;
- **WPP** — класс языков $L \subseteq \Sigma^*$, для которых существуют функции $f \in \mathbf{GapP}$ и $g \in \mathbf{FP}$, такие, что для любого $x \in \Sigma^*$ значение функции $g(x)$ отлично от нуля и $x \in L \Rightarrow f(x) = g(x)$ и $x \notin L \Rightarrow f(x) = 0$.

Для этих классов справедливы следующие важные соотношения [63]:

$$\mathbf{P} \subseteq \mathbf{UP} \subseteq \mathbf{SPP} \subseteq \mathbf{WPP} \subseteq \mathbf{C}_{=}\mathbf{P} \subseteq \mathbf{PP},$$
$$\mathbf{co} - \mathbf{NP} \subseteq \mathbf{C}_{=}\mathbf{P},$$
$$\mathbf{NP} \subseteq \mathbf{co} - \mathbf{C}_{=}\mathbf{P} = \mathbf{NQP},$$
$$\mathbf{P} \subseteq \mathbf{EQP} \subseteq \mathbf{BQP} \subseteq \mathbf{WPP} \subseteq \mathbf{co} - \mathbf{C}_{=}\mathbf{P}.$$

Кроме того, в работе [67] доказано, что $\mathbf{PP} = \mathbf{PostBQP}$.

Исходя из определений, очевидными аналогами классов **NP** и **UP** представляются классы **PP** и **SPP** соответственно. При этом классы **PP** и $\mathbf{C}_{=}\mathbf{P}$ размещают в иерархии классических классов вычислительной сложности на верхней границе квантовых вычислений, а класс **WPP** на сегодняшний день определяет безопасную границу для криптографии. Соответственно класс **SPP** является «плохим» аналогом для **UP**: перспективные криптографические алгоритмы желательно строить на базе **WPP** или даже $\mathbf{C}_{=}\mathbf{P}$.

Следует отметить, что с теоретической точки зрения нет фундаментальных препятствий для построения криптографических алгоритмов на базе столь сложных клас-

сов, какими являются классы **WPP** и $\mathbf{C}_{=}\mathbf{P}$. Рассмотрим эквивалентное определение класса **PP**, не использующее класс **GapP**: **PP** — класс языков $L \subseteq \Sigma^*$, для которых существуют функции $f \in \#\mathbf{P}$ и $g \in \mathbf{FP}$, такие, что $x \in L \Leftrightarrow f(x) > g(x)$ для любого $x \in \Sigma^*$ [68]. Классические криптографические схемы предполагают, что легальные участники передачи информации A и B, зная некоторую секретную информацию K, могут легко решить задачу $f(x) > 0$, а для злоумышленника C, не имеющего доступа к информации K, решение задачи $f(x) > 0$ является трудным. В случае класса **PP** участники A и B могут выбирать функции $g \in \mathbf{FP} \subseteq \#\mathbf{P}$, удобные для отображения $f(x) - g(x) \to f'(x) \in \#\mathbf{P}$. В результате, зная некоторую секретную информацию K и функцию $g(x)$, A и B могут решить задачу $f'(x) > 0$, сложность которой сравнима со сложностью задачи $f(x) > 0$, т. е. находится на уровне **NP**. Злоумышленник C, не имеющий доступа к информации K, должен решать задачу $f(x) > g(x)$, сложность которой находится на уровне **PP**, или суметь демаскировать $f'(x)$ по $f(x)$ и $g(x)$. Таким образом, мы имеем ситуацию, которая с точки зрения вычислительной сложности совершенно аналогична схеме, предложенной в [2] и предполагающей, что злоумышленник либо должен решать **NP**-трудную задачу декодирования, либо демаскировать исходную матрицу G по матрице $G' = SGP$.

Естественно, между отсутствием фундаментальных препятствий и практической реализацией всегда имеется значительный разрыв, требующий построения соответствующей теории. Однако некоторую перспективу для разработки практического подхода для построения криптографических алгоритмов на базе класса **PP** можно проиллюстрировать на основе известных результатов для следующей проблемы.

Majority satisfiability (MajSAT)

Дано: Булева функция $f(x_1, x_2, \ldots, x_m)$.

Вопрос: Верно ли, что не менее половины наборов значений $x_j \in \{0, 1\}$, $1 \leqslant j \leqslant m$, позволяют получить $f(x_1, x_2, \ldots, x_m) = 1$?

В общем случае проблема MajSAT является **PP**-полной [69]. Если вместо произвольной булевой функции $f(x_1, x_2, \ldots, x_m)$ рассматривать 3-КНФ, то проблема MajSAT разрешима за полиномиальное время [70]. Кроме того, известны варианты MajSAT, которые являются **PP**-полными при ограничении функции $f(x_1, x_2, \ldots, x_m)$ на случай 3-КНФ, но требуют другой доли в общем количестве решений [71]. Таким образом, легальные участники передачи информации A и B могут представлять открытый текст в виде исходных данных проблемы MajSAT для 3-КНФ. Шифрование будет заключаться в преобразовании 3-КНФ в булеву функцию произвольного вида или другую 3-КНФ с другой долей решений. Хороший фундамент для маскировки булевых функций может предоставить шифрование логики [72–74], а преобразования, позволяющие изменять или сохранять долю решений, найти не представляет труда. Например, для произвольной булевой функции $f(x_1, x_2, \ldots, x_m)$ булева функция

$$(f(x_1, x_2, \ldots, x_m) \wedge x_{m+1}) \vee (f(x_1, x_2, \ldots, x_m) \wedge \neg x_{m+1})$$

имеет ровно в 2 раза больше решений, чем исходная.

Рассмотрим ряд непосредственных следствий из теоремы 1, позволяющих установить некоторые аналоги проблемы MLD для классов вычислительной сложности за пределами класса **NP** и представляющих интерес с точки зрения постквантовой криптографии. Сформулируем следующие проблемы.

DiffPM$_{=0}$

Дано: Графы $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$.

Вопрос: Верно ли, что количество множеств $M_1 \subseteq E_1$, таких, что каждая вершина графа G_1 инцидентна ровно одному ребру из множества M_1, равно количеству множеств $M_2 \subseteq E_2$, таких, что каждая вершина графа G_2 инцидентна ровно одному ребру из множества M_2?

DiffPM$_{>0}$

Дано: Графы $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$.

Вопрос: Верно ли, что количество множеств $M_1 \subseteq E_1$, таких, что каждая вершина графа G_1 инцидентна ровно одному ребру из множества M_1, больше количества множеств $M_2 \subseteq E_2$, таких, что каждая вершина графа G_2 инцидентна ровно одному ребру из множества M_2?

DiffPM$_{=g}$

Дано: Графы $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$, натуральное число k.

Обещание: Пусть X — количество множеств $M_1 \subseteq E_1$, таких, что каждая вершина графа G_1 инцидентна ровно одному ребру из множества M_1, Y — количество множеств $M_2 \subseteq E_2$, таких, что каждая вершина графа G_2 инцидентна ровно одному ребру из множества M_2. Известно, что $X = Y$ или $X = Y + k$.

Вопрос: Верно ли, что $X = Y + k$?

DiffMLD$_{=0}$

Дано: Двоичные матрицы $H_1 \in \mathbb{Z}_2^{m_1 \times n_1}$, $H_2 \in \mathbb{Z}_2^{m_1 \times n_1}$, векторы $s_1 \in \mathbb{Z}_2^{m_1}$, $s_2 \in \mathbb{Z}_2^{m_2}$, натуральные числа k_1 и k_2.

Вопрос: Верно ли, что количество попарно различных векторов $x \in \mathbb{Z}_2^{n_1}$, таких, что $H_1 x = s_1$ и $\mathrm{wt}(x) \leqslant k_1$, равно количеству попарно различных векторов $y \in \mathbb{Z}_2^{n_2}$, таких, что $H_2 y = s_2$ и $\mathrm{wt}(y)) \leqslant k_2$?

DiffMLD$_{>0}$

Дано: Двоичные матрицы $H_1 \in \mathbb{Z}_2^{m_1 \times n_1}$, $H_2 \in \mathbb{Z}_2^{m_1 \times n_1}$, векторы $s_1 \in \mathbb{Z}_2^{m_1}$, $s_2 \in \mathbb{Z}_2^{m_2}$, натуральные числа k_1 и k_2.

Вопрос: Верно ли, что количество попарно различных векторов $x \in \mathbb{Z}_2^{n_1}$, таких, что $H_1 x = s_1$ и $\mathrm{wt}(x) \leqslant k_1$, больше количества попарно различных векторов $y \in \mathbb{Z}_2^{n_2}$, таких, что $H_2 y = s_2$ и $\mathrm{wt}(y) \leqslant k_2$?

DiffMLD$_{=g}$

Дано: Двоичные матрицы $H_1 \in \mathbb{Z}_2^{m_1 \times n_1}$, $H_2 \in \mathbb{Z}_2^{m_1 \times n_1}$, векторы $s_1 \in \mathbb{Z}_2^{m_1}$, $s_2 \in \mathbb{Z}_2^{m_2}$, натуральные числа k_1, k_2, k.

Обещание: Пусть X — количество попарно различных векторов $x \in \mathbb{Z}_2^{n_1}$, таких, что $H_1 x = s_1$ и $\mathrm{wt}(x) \leqslant k_1$, Y — количество попарно различных векторов $y \in \mathbb{Z}_2^{n_2}$, таких, что $H_2 y = s_2$ и $\mathrm{wt}(y) \leqslant k_2$. Известно, что $X = Y$ или $X = Y + k$.

Вопрос: Верно ли, что $X = Y + k$?

Заметим, что в отличие от большинства классов вычислительной сложности, для определения которых достаточно использования одной функции f, определение класса **WPP** содержит функции f и g, что затрудняет формулировку естественных проблем для класса **WPP** при помощи традиционного определения того, что дано, и вопроса. Обычно формулировка проблемы для класса **WPP** включает не только исходные данные и вопрос, но и некоторое обещание выполнимости какого-то условия. Предполагается, что алгоритм, решающий проблему, получает на вход исходные данные. Алгоритм не знает, выполняется ли условие, содержащееся в обещании, для этих исходных данных, и не проверяет его выполнимость. Алгоритм, решающий проблему,

должен гарантировать правильность ответа, если условие выполняется. Если условие не выполняется, то алгоритм может выдать ошибочный ответ. В частности, алгоритм, решающий проблему DiffMLD$_{=g}$, должен ориентироваться на обещание того, что $X = Y$ или $X = Y + k$, и ответить на вопрос о том, верно ли, что $X = Y + k$. Получив на вход исходные данные, алгоритм должен выдать ответ «Да», если $X = Y + k$; ответ «Нет», если $X = Y$; если ни одно из равенств $X = Y$ и $X = Y + k$ не выполняется, то алгоритм может выдать любой из ответов «Да» и «Нет» или вообще не выдать ответа, продолжая работать вечно.

В [75] доказано, что проблема DiffPM$_{=0}$ является $\mathbf{C_{=}P}$-полной. В работе [76] для проблем DiffPM$_{>0}$ и DiffPM$_{=g}$ показана полнота в классах **PP** и **WPP** соответственно. Из этих результатов и теоремы 1 непосредственно вытекает

Следствие 2. Проблемы DiffMLD$_{=g}$, DiffMLD$_{=0}$ и DiffMLD$_{>0}$ являются полными в классах **WPP**, $\mathbf{C_{=}P}$ и **PP** соответственно.

Заключение

В работе рассмотрен количественный аналог #MLD проблемы декодирования по принципу максимального правдоподобия MLD. Для проблемы MLD установлена экономная сводимость от проблемы совершенного паросочетания и слабо экономная сводимость от проблемы максимального разреза. Как следствие, мы получили #**P**-полноту проблемы #MLD. Кроме того, это позволило сформулировать вычислительно трудные аналоги проблемы декодирования по принципу максимального правдоподобия для классов вычислительной сложности, представляющих интерес с точки зрения постквантовой криптографии. В частности, доказана полнота проблем DiffMLD$_{=g}$, DiffMLD$_{=0}$ и DiffMLD$_{>0}$ в классах **WPP**, $\mathbf{C_{=}P}$ и **PP** соответственно. Кроме того, получено альтернативное доказательство **NP**-полноты проблемы MLD, что является дополнительным обоснованием надежности криптографических алгоритмов на основе Classic McEliece.

ЛИТЕРАТУРА

1. *Berlekamp E., McEliece R., and van Tilborg H.* On the inherent intractability of certain coding problems (Corresp.) // IEEE Trans. Inform. Theory. 1978. V. 24. No. 3. P. 384–386.
2. *McEliece R. J.* A public-key cryptosystem based on algebraic coding theory // Deep Space Network Progress Report. 1978. V. 42. No. 44. P. 114–116.
3. *Kichna A. and Farchane A.* A survey on various decoding algorithms for McEliece cryptosystem based on QC-MDPC codes // Proc. IRASET. Mohammedia, Morocco, 2023. P. 1–7.
4. *Liu J., Tong X., Wang Z., et al.* An improved McEliece cryptosystem based on QC-MDPC code with compact key size // Telecommun. Syst. 2022. V. 80. P. 17–32.
5. *Lau T. and Tan C.* On the design and security of Lee metric McEliece cryptosystems // Des. Codes Cryptogr. 2022. V. 90. No. 3. P. 695–717.
6. `https://www.nist.gov/programs-projects/post-quantum-cryptography`. 2024.
7. `https://classic.mceliece.org/`. 2024.
8. `https://web.archive.org/web/20171229103229/https://nts-kem.io/`. 2024.
9. *Niederreiter H.* Knapsack-type cryptosystems and algebraic coding theory // Problems Control Inform. Theory. 1986. V. 15. No. 2. P. 159–166.
10. *Dent A. W.* A designer's guide to KEMs // LNCS. 2003. V. 2898. P. 133–151.
11. *Fujisaki E. and Okamoto T.* Secure integration of asymmetric and symmetric encryption schemes // J. Cryptology. 2013. V. 26. No. 1. P. 80–101.

12. *Fujita H.* Quantum McEliece public-key cryptosystem // Quantum Inform. & Comput. 2012. V. 12. No. 3–4. P. 181–202.
13. *Oh Y., Jang K., Lim S., et al.* Quantum implementation of core operations in classic McEliece // Proc. PlatCon. Busan, Korea, 2023. P. 67–72.
14. *Fuentes P., Martinez J. E., Crespo P. M., and Garcia-Frias J.* Degeneracy and its impact on the decoding of sparse quantum codes // IEEE Access. 2021. V. 9. P. 89093–89119.
15. *Kubica A. and Vasmer M.* Single-shot quantum error correction with the three-dimensional subsystem toric code // Nat. Commun. 2022. V. 13. Article No. 6272.
16. *Kuo K.-Y. and Lai C.-Y.* Exploiting degeneracy in belief propagation decoding of quantum codes // npj Quantum Inform. 2022. V. 8. Article No. 111.
17. *Elbro F. and Majenz C.* An algebraic attack against McEliece-like cryptosystems based on BCH codes // Proc. ITW. Saint-Malo, France, 2023. P. 70–75.
18. *Gray H., Battarbee C., Shahandashti S. F., and Kahrobaei D.* A novel attack on McEliece's cryptosystem // Intern. J. Computer Math.: Computer Systems Theory. 2023. V. 8. No. 3. P. 178–191.
19. *Kirshanova E. and May A.* Breaking Goppa-based McEliece with hints // Inform. Comput. 2023. V. 293. Article No. 105045.
20. *Baldi M., Bianchi M., and Chiaraluce F.* Security and complexity of the McEliece cryptosystem based on quasi-cyclic low-density parity-check codes // IET Inform. Security. 2013. V. 7. No. 3. P. 212–220.
21. *Freudenberger J. and Thiers J. P.* A new class of Q-ary codes for the McEliece cryptosystem // Cryptography. 2021. V. 5. No. 1. Article No. 11.
22. *Mariot L., Picek S., and Yorgova R.* On McEliece-type cryptosystems using self-dual codes with large minimum weight // IEEE Access. 2023. V. 11. P. 43511–43519.
23. *Hsieh M.-H. and Le Gall F.* NP-hardness of decoding quantum error-correction codes // Phys. Rev. A. 2011. V. 83. Article No. 052331.
24. *Kuo K.-Y. and Lu C.-C.* On the hardness of decoding quantum stabilizer codes under the depolarizing channel // Intern. Symp. Inform. Theory and its Appl. Honolulu, USA, 2012. P. 208–211.
25. *Kuo K.-Y. and Lu C.-C.* On the hardnesses of several quantum decoding problems // Quantum Inf. Process. 2020. V. 19. Article No. 123.
26. *Iyer P. and Poulin D.* Hardness of decoding quantum stabilizer codes // IEEE Trans. Inform. Theory. 2015. V. 61. No. 9. P. 5209–5223.
27. *Kuo K.-Y. and Lai C.-Y.* The encoding and decoding complexities of entanglement-assisted quantum stabilizer codes // Proc ISIT. Paris, France, 2019. P. 2893–2897
28. *Chamberland C., Goncalves L., Sivarajah P., et al.* Techniques for combining fast local decoders with global decoders under circuit-level noise // Quantum Sci. Technology. 2023. V. 8. Article No. 045011.
29. *Hammar K., Orekhov A., Hybelius P. W., et al.* Error-rate-agnostic decoding of topological stabilizer codes // Phys. Rev. A. 2022. V. 105. Article No. 042616.
30. *Théveniaut H. and van Nieuwenburg E.* A NEAT quantum error decoder // SciPost Physics. 2021. V. 11. Article No. 005.
31. *Colomer L. D., Skotiniotis M., and Muñoz-Tapia R.* Reinforcement learning for optimal error correction of toric codes // Phys. Let. A. 2020. V. 384. No. 17. Article 126353.
32. *Sweke R., Kesselring M. S., van Nieuwenburg E. P. L., and Eisert J.* Reinforcement learning decoders for fault-tolerant quantum computation // Mach. Learn.: Sci. Technol. 2021. V. 2. Article No. 025005.

33. *Krastanov S. and Jiang L.* Deep neural network probabilistic decoder for stabilizer codes // Sci. Rep. 2017. V. 7. Article No. 11003.
34. *Baireuther P., Caio M. D., Criger B., et al.* Neural network decoder for topological color codes with circuit level noise // New J. Physics. 2019. V. 21. Article No. 013003.
35. *Gicev S., Hollenberg L. C. L., and Usman M.* A scalable and fast artificial neural network syndrome decoder for surface codes // Quantum. 2023. V. 7. Article No. 1058.
36. *Bordoni S. and Giagu S.* Convolutional neural network based decoders for surface codes // Quantum Inf. Process. 2023. V. 22. Article No. 151.
37. *Li A., Li F., Gan Q., and Ma H.* Convolutional-Neural-Network-Based hexagonal quantum error correction decoder // Appl. Sci. 2023. V. 13. No. 17. Article 9689.
38. *Wenger E., Chen M., Charton F., and Lauter K.* SALSA: Attacking Lattice Cryptography with Transformers. Cryptology ePrint Archive. 2022. Paper 2022/935. `https://eprint.iacr.org/2022/935`.
39. *Bruck J. and Naor M.* The hardness of decoding linear codes with preprocessing // IEEE Trans. Inform. Theory. 1990. V. 36. No. 2. P. 381–385.
40. *Dumer I., Micciancio D., and Sudan M.* Hardness of approximating the minimum distance of a linear code // IEEE Trans. Inform. Theory. 2003. V. 49. No. 1. P. 22–37.
41. *Vardy A.* The intractability of computing the minimum distance of a code // IEEE Trans. Inform. Theory. 1997. V. 43. No. 6. P. 1757–1766.
42. *Papadimitriou C. H.* Computational Complexity. Reading, Addison-Wesley, 1994. 523 p.
43. *Juban L.* Dichotomy theorem for the generalized unique satisfiability problem // LNCS. 1999. V. 1684. P. 327–337.
44. *Creignou N. and Hermann M.* Complexity of generalized satisfiability counting problems // Inform. Comput. 1996. V. 125. No. 1. P. 1–12.
45. *Valiant L. G.* The complexity of computing the permanent // Theoretical Computer Sci. 1979. V. 8. No. 2. P. 189–201.
46. *Barbanchon R.* On unique graph 3-colorability and parsimonious reductions in the plane // Theoretical Computer Sci. 2004. V. 319. No. 1–3. P. 455–482.
47. *Karp R. M.* Reducibility among combinatorial problems // R. E. Miller, J. W. Thatcher, and J. D. Bohlinger (eds.). Complexity of Computer Computations. The IBM Research Symposia Series. Boston: Springer, 1972. P. 85–103.
48. *Garey M. R. and Johnson D. S.* Computers and Intractability: A Guide to the Theory of NP-Completeness. San Francisco: Freeman, 1979. 338 p.
49. *Garey M. R., Johnson D. S., and Stockmeyer L.* Some simplified NP-complete graph problems // Theoretical Computer Sci. 1976. V. 1. No. 3. P. 237–267.
50. *Bruck J. and Blaum M.* Neural networks, error-correcting codes, and polynomials over the binary n-cube // IEEE Trans. Inform. Theory. 1989. V. 35. No. 5. P. 976–987.
51. *Барг С.* Некоторые новые NP-полные задачи кодирования // Проблемы передачи информации. 1994. Т. 30. № 3. С. 23–28.
52. *Jelínková E., Suchý O., Hliněný P., and Kratochvíl J.* Parameterized problems related to Seidel's switching // Discrete Math. Theoretical Computer Sci. 2011. V. 13. No. 2. P. 19–42.
53. *Tanatmis A.* Mathematical Programming Approaches for Decoding of Binary Linear Codes. Vom Fachbereich Mathematik der Technischen Universität Kaiserslautern zur Erlangung des akademischen Grades Doktor der Naturwissenschaften (Doctor rerum naturalium, Dr. rer. nat.) genehmigte Dissertation, 2011.
54. *Arora S., Babai L., Stern J., and Sweedyk Z.* The hardness of approximate optima in lattices, codes, and systems of linear equations // J. Computer System Sci. 1997. V. 54. No. 2. P. 317–331.

55. *Lindell Y.* Introduction to Coding Theory. https://yehudalindell.com/teaching/introduction-to-coding-theory/. 2024.
56. *Kaye P., Laflamme R., and Mosca M.* An Introduction to Quantum Computing. N.Y.: Oxford University Press, 2007. 274 p.
57. *Nielsen M. A. and Chuang I. L.* Quantum Computation and Quantum Information. Cambridge: Cambridge University Press, 2010. 676 p.
58. *Ceselli A. and Premoli M.* On good encodings for quantum annealer and digital optimization solvers // Sci. Rep. 2023. V. 13. Article No. 5628.
59. *Koch D., Cutugno M., Patel S., et al.* Variational amplitude amplification for solving QUBO problems // Quantum Rep. 2023. V. 5. No. 4. P. 625–658.
60. *Pokharel B., Izquierdo Z. G., Lott P. A., et al.* Inter-generational comparison of quantum annealers in solving hard scheduling problems // Quantum Inf. Process. 2023. V. 22. Article No. 364.
61. *Aggarwal D., Bennett H., Brakerski Z., et al.* Lattice problems beyond polynomial time // Proc. STOC'2023. Orlando, FL, USA, 2023. P. 1516–1526.
62. *Pass R., Tseng W. L. D., and Venkitasubramaniam M.* Towards non-black-box lower bounds in cryptography // LNCS. 2011. V. 6597. P. 579–596.
63. *Spakowski H., Thakur M., and Tripathi R.* Quantum and classical complexity classes: Separations, collapses, and closure properties // Inform. Comput. 2005. V. 200. No. 1. P. 1–34.
64. *Adleman L., DeMarrais J., and Huang M.* Quantum computability // SIAM J. Computing. 1997. V. 26. No. 5. P. 1524–1540.
65. *Bernstein E. and Vazirani U.* Quantum complexity theory // SIAM J. Computing. 1997. V. 26. No. 5. P. 1411–1473.
66. *Shor P. W.* Algorithms for quantum computation: discrete logarithms and factoring // Proc. 35th Ann. Symp. FOCS. Santa Fe, USA, 1994. P. 124–134.
67. *Aaronson S.* Quantum computing, postselection, and probabilistic polynomial-time // Proc. R. Soc. A. 2005. V. 461. P. 3473–3482.
68. *Ogiwara M. and Hemachandra L. A.* A complexity theory for feasible closure properties // J. Computer System Sci. 1993. V. 46. No. 3. P. 295–325.
69. *Gill J.* Computational complexity of probabilistic Turing machines // SIAM J. Computing. 1977. V. 6. No. 4. P. 675–695.
70. *Akmal S. and Williams R.* MAJORITY-3SAT (and related problems) in polynomial time // Proc. 62nd Ann. Symp. FOCS. Denver, USA, 2022. P. 1033–1043.
71. *Bailey D. D., Dalmau V., and Kolaitis P. G.* Phase transitions of PP-complete satisfiability problems // Discrete Appl. Math. 2007. V. 155. No. 12. P. 1627–1639.
72. *Chandra S. S., Kannan R. J., Balaji B. S., et al.* Efficient design and analysis of secure CMOS logic through logic encryption // Sci. Rep. 2023. V. 13. Article No. 1145.
73. *Liang J., Wang K., Xi W., et al.* SILL: Preventing structural attack for logic locking // IEICE Electronics Express. 2023. V. 20. No. 2. P. 1–6.
74. *Rajendran J. and Garg S.* Logic encryption // Forte D., Bhunia S., and Tehranipoor M. M. (eds.). Hardware Protection through Obfuscation. Cham: Springer, 2017. P. 71–88.
75. *Curticapean R.* The simple, little and slow things count: On parameterized counting complexity. Dissertation for Obtaining the Title of Doctor rerum naturalium (Dr. rer. nat) of the Faculties of Natural Sciences and Technology of Saarland University, Saarbrücken, 2015.
76. *Bakali E., Chalki A., Kanellopoulos S., et al.* On the Power of Counting the Total Number of Computation Paths of NPTMs. https://arxiv.org/abs/2306.11614. 2024.

REFERENCES

1. *Berlekamp E., McEliece R., and van Tilborg H.* On the inherent intractability of certain coding problems (Corresp.). IEEE Trans. Inform. Theory, 1978, vol. 24, no. 3, pp. 384–386.
2. *McEliece R. J.* A public-key cryptosystem based on algebraic coding theory. Deep Space Network Progress Report, 1978, vol. 42, no. 44, pp. 114–116.
3. *Kichna A. and Farchane A.* A survey on various decoding algorithms for McEliece cryptosystem based on QC-MDPC codes. Proc. IRASET, Mohammedia, Morocco, 2023, pp. 1–7.
4. *Liu J., Tong X., Wang Z., et al.* An improved McEliece cryptosystem based on QC-MDPC code with compact key size. Telecommun. Syst., 2022, vol. 80, pp. 17–32.
5. *Lau T. and Tan C.* On the design and security of Lee metric McEliece cryptosystems. Des. Codes Cryptogr., 2022, vol. 90, no. 3, pp. 695–717.
6. `https://www.nist.gov/programs-projects/post-quantum-cryptography`. 2024.
7. `https://classic.mceliece.org/`. 2024.
8. `https://web.archive.org/web/20171229103229/https://nts-kem.io/`. 2024.
9. *Niederreiter H.* Knapsack-type cryptosystems and algebraic coding theory. Problems Control Inform. Theory, 1986, vol. 15, no. 2, pp. 159–166.
10. *Dent A. W.* A designer's guide to KEMs. LNCS, 2003, vol. 2898, pp. 133–151.
11. *Fujisaki E. and Okamoto T.* Secure integration of asymmetric and symmetric encryption schemes. J. Cryptology, 2013, vol. 26, no. 1, pp. 80–101.
12. *Fujita H.* Quantum McEliece public-key cryptosystem. Quantum Inform. & Comput., 2012, vol. 12, no. 3–4, pp. 181–202.
13. *Oh Y., Jang K., Lim S., et al.* Quantum implementation of core operations in classic McEliece. Proc. PlatCon, Busan, Korea, 2023, pp. 67–72.
14. *Fuentes P., Martinez J. E., Crespo P. M., and Garcia-Frias J.* Degeneracy and its impact on the decoding of sparse quantum codes. IEEE Access, 2021, vol. 9, pp. 89093–89119.
15. *Kubica A. and Vasmer M.* Single-shot quantum error correction with the three-dimensional subsystem toric code. Nat. Commun., 2022, vol. 13, article no. 6272.
16. *Kuo K.-Y. and Lai C.-Y.* Exploiting degeneracy in belief propagation decoding of quantum codes. npj Quantum Inform., 2022, vol. 8, article no. 111.
17. *Elbro F. and Majenz C.* An algebraic attack against McEliece-like cryptosystems based on BCH codes. Proc. ITW, Saint-Malo, France, 2023, pp. 70–75.
18. *Gray H., Battarbee C., Shahandashti S. F., and Kahrobaei D.* A novel attack on McEliece's cryptosystem. Intern. J. Computer Math.: Computer Systems Theory, 2023, vol. 8, no. 3, pp. 178–191.
19. *Kirshanova E. and May A.* Breaking Goppa-based McEliece with hints. Inform. Comput., 2023, vol. 293, article no. 105045.
20. *Baldi M., Bianchi M., and Chiaraluce F.* Security and complexity of the McEliece cryptosystem based on quasi-cyclic low-density parity-check codes. IET Inform. Security, 2013, vol. 7, no. 3, pp. 212–220.
21. *Freudenberger J. and Thiers J. P.* A new class of Q-ary codes for the McEliece cryptosystem. Cryptography, 2021, vol. 5, no. 1, article no. 11.
22. *Mariot L., Picek S., and Yorgova R.* On McEliece-type cryptosystems using self-dual codes with large minimum weight. IEEE Access, 2023, vol. 11, pp. 43511–43519.
23. *Hsieh M.-H. and Le Gall F.* NP-hardness of decoding quantum error-correction codes. Phys. Rev. A, 2011, vol. 83, article no. 052331.

24. *Kuo K.-Y. and Lu C.-C.* On the hardness of decoding quantum stabilizer codes under the depolarizing channel. Intern. Symp. Inform. Theory and its Appl., Honolulu, USA, 2012, pp. 208–211.
25. *Kuo K.-Y. and Lu C.-C.* On the hardnesses of several quantum decoding problems. Quantum Inf. Process., 2020, vol. 19, article no. 123.
26. *Iyer P. and Poulin D.* Hardness of decoding quantum stabilizer codes. IEEE Trans. Inform. Theory, 2015, vol. 61, no. 9, pp. 5209–5223.
27. *Kuo K.-Y. and Lai C.-Y.* The encoding and decoding complexities of entanglement-assisted quantum stabilizer codes. Proc ISIT, Paris, France, 2019, pp. 2893–2897
28. *Chamberland C., Goncalves L., Sivarajah P., et al.* Techniques for combining fast local decoders with global decoders under circuit-level noise. Quantum Sci. Technology, 2023, vol. 8, article no. 045011.
29. *Hammar K., Orekhov A., Hybelius P. W., et al.* Error-rate-agnostic decoding of topological stabilizer codes. Phys. Rev. A, 2022, vol. 105, article no. 042616.
30. *Théveniaut H. and van Nieuwenburg E.* A NEAT quantum error decoder. SciPost Physics, 2021, vol. 11, article no. 005.
31. *Colomer L. D., Skotiniotis M., and Muñoz-Tapia R.* Reinforcement learning for optimal error correction of toric codes. Phys. Let. A, 2020, vol. 384, no. 17. Article 126353.
32. *Sweke R., Kesselring M. S., van Nieuwenburg E. P. L., and Eisert J.* Reinforcement learning decoders for fault-tolerant quantum computation. Mach. Learn.: Sci. Technol, 2021, vol. 2, article no. 025005.
33. *Krastanov S. and Jiang L.* Deep neural network probabilistic decoder for stabilizer codes. Sci. Rep., 2017, vol. 7, article no. 11003.
34. *Baireuther P., Caio M. D., Criger B., et al.* Neural network decoder for topological color codes with circuit level noise. New J. Physics, 2019, vol. 21, article no. 013003.
35. *Gicev S., Hollenberg L. C. L., and Usman M.* A scalable and fast artificial neural network syndrome decoder for surface codes. Quantum, 2023, vol. 7, article no. 1058.
36. *Bordoni S. and Giagu S.* Convolutional neural network based decoders for surface codes. Quantum Inf. Process., 2023, vol. 22, article no. 151.
37. *Li A., Li F., Gan Q., and Ma H.* Convolutional-Neural-Network-Based hexagonal quantum error correction decoder. Appl. Sci., 2023, vol. 13, no. 17, article 9689.
38. *Wenger E., Chen M., Charton F., and Lauter K.* SALSA: Attacking Lattice Cryptography with Transformers. Cryptology ePrint Archive, 2022, paper 2022/935, `https://eprint.iacr.org/2022/935`.
39. *Bruck J. and Naor M.* The hardness of decoding linear codes with preprocessing. IEEE Trans. Inform. Theory. 1990, vol. 36, no. 2, pp. 381–385.
40. *Dumer I., Micciancio D., and Sudan M.* Hardness of approximating the minimum distance of a linear code. IEEE Trans. Inform. Theory, 2003, vol. 49, no. 1, pp. 22–37.
41. *Vardy A.* The intractability of computing the minimum distance of a code. IEEE Trans. Inform. Theory, 1997, vol. 43, no. 6, pp. 1757–1766.
42. *Papadimitriou C. H.* Computational Complexity. Reading, Addison-Wesley, 1994. 523 p.
43. *Juban L.* Dichotomy theorem for the generalized unique satisfiability problem. LNCS, 1999, vol. 1684, pp. 327–337.
44. *Creignou N. and Hermann M.* Complexity of generalized satisfiability counting problems. Inform. Comput., 1996, vol. 125, no. 1, pp. 1–12.
45. *Valiant L. G.* The complexity of computing the permanent. Theoretical Computer Sci., 1979, vol. 8, no. 2, pp. 189–201.

46. *Barbanchon R.* On unique graph 3-colorability and parsimonious reductions in the plane. Theoretical Computer Sci., 2004, vol. 319, no. 1–3, pp. 455–482.
47. *Karp R. M.* Reducibility among combinatorial problems. R. E. Miller, J. W. Thatcher, and J. D. Bohlinger (eds.). Complexity of Computer Computations. The IBM Research Symposia Series, Boston, Springer, 1972, pp. 85–103.
48. *Garey M. R. and Johnson D. S.* Computers and Intractability: A Guide to the Theory of NP-Completeness. San Francisco, Freeman, 1979, 338 p.
49. *Garey M. R., Johnson D. S., and Stockmeyer L.* Some simplified NP-complete graph problems. Theoretical Computer Sci., 1976, vol. 1, no. 3, pp. 237–267.
50. *Bruck J. and Blaum M.* Neural networks, error-correcting codes, and polynomials over the binary n-cube. IEEE Trans. Inform. Theory, 1989, vol. 35, no. 5, pp. 976–987.
51. *Barg S.* Nekotorye novye NP-polnye zadachi kodirovaniya [Some new NP-complete coding problems]. Problemy Peredachi Informatsii, 1994, vol. 30, no. 3, pp. 23–28. (in Russian)
52. *Jelínková E., Suchý O., Hliněný P., and Kratochvíl J.* Parameterized problems related to Seidel's switching. Discrete Math. Theoretical Computer Sci., 2011, vol. 13, no. 2, pp. 19–42.
53. *Tanatmis A.* Mathematical Programming Approaches for Decoding of Binary Linear Codes. Vom Fachbereich Mathematik der Technischen Universität Kaiserslautern zur Erlangung des akademischen Grades Doktor der Naturwissenschaften (Doctor rerum naturalium, Dr. rer. nat.) genehmigte Dissertation, 2011.
54. *Arora S., Babai L., Stern J., and Sweedyk Z.* The hardness of approximate optima in lattices, codes, and systems of linear equations. J. Computer System Sci., 1997, vol. 54, no. 2, pp. 317–331.
55. *Lindell Y.* Introduction to Coding Theory. `https://yehudalindell.com/teaching/introduction-to-coding-theory/`, 2024.
56. *Kaye P., Laflamme R., and Mosca M.* An Introduction to Quantum Computing. N.Y., Oxford University Press, 2007, 274 p.
57. *Nielsen M. A. and Chuang I. L.* Quantum Computation and Quantum Information. Cambridge, Cambridge University Press, 2010, 676 p.
58. *Ceselli A. and Premoli M.* On good encodings for quantum annealer and digital optimization solvers. Sci. Rep., 2023, vol. 13, article no. 5628.
59. *Koch D., Cutugno M., Patel S., et al.* Variational amplitude amplification for solving QUBO problems. Quantum Rep., 2023, vol. 5, no. 4, pp. 625–658.
60. *Pokharel B., Izquierdo Z. G., Lott P. A., et al.* Inter-generational comparison of quantum annealers in solving hard scheduling problems. Quantum Inf. Process., 2023, vol. 22, article no. 364.
61. *Aggarwal D., Bennett H., Brakerski Z., et al.* Lattice problems beyond polynomial time. Proc. STOC'2023, Orlando, FL, USA, 2023, pp. 1516–1526.
62. *Pass R., Tseng W. L. D., and Venkitasubramaniam M.* Towards non-black-box lower bounds in cryptography. LNCS, 2011, vol. 6597, pp. 579–596.
63. *Spakowski H., Thakur M., and Tripathi R.* Quantum and classical complexity classes: Separations, collapses, and closure properties. Inform. Comput., 2005, vol. 200, no. 1, pp. 1–34.
64. *Adleman L., DeMarrais J., and Huang M.* Quantum computability. SIAM J. Computing, 1997, vol. 26, no. 5, pp. 1524–1540.
65. *Bernstein E. and Vazirani U.* Quantum complexity theory. SIAM J. Computing, 1997, vol. 26, no. 5, pp. 1411–1473.
66. *Shor P. W.* Algorithms for quantum computation: discrete logarithms and factoring. Proc. 35th Ann. Symp. FOCS, Santa Fe, USA, 1994, pp. 124–134.

67. *Aaronson S.* Quantum computing, postselection, and probabilistic polynomial-time. Proc. R. Soc. A, 2005, vol. 461, pp. 3473–3482.
68. *Ogiwara M. and Hemachandra L. A.* A complexity theory for feasible closure properties. J. Computer System Sci., 1993, vol. 46, no. 3, pp. 295–325.
69. *Gill J.* Computational complexity of probabilistic Turing machines. SIAM J. Computing. 1977, vol. 6, no. 4, pp. 675–695.
70. *Akmal S. and Williams R.* MAJORITY-3SAT (and related problems) in polynomial time. Proc. 62nd Ann. Symp. FOCS, Denver, USA, 2022, pp. 1033–1043.
71. *Bailey D. D., Dalmau V., and Kolaitis P. G.* Phase transitions of PP-complete satisfiability problems. Discrete Appl. Math., 2007, vol. 155, no. 12, pp. 1627–1639.
72. *Chandra S. S., Kannan R. J., Balaji B. S., et al.* Efficient design and analysis of secure CMOS logic through logic encryption. Sci. Rep., 2023, vol. 13, article no. 1145.
73. *Liang J., Wang K., Xi W., et al.* SILL: Preventing structural attack for logic locking. IEICE Electronics Express, 2023, vol. 20, no. 2, pp. 1–6.
74. *Rajendran J. and Garg S.* Logic encryption. Forte D., Bhunia S., and Tehranipoor M. M. (eds.). Hardware Protection through Obfuscation. Cham, Springer, 2017, pp. 71–88.
75. *Curticapean R.* The simple, little and slow things count: On parameterized counting complexity. Dissertation for Obtaining the Title of Doctor rerum naturalium (Dr. rer. nat) of the Faculties of Natural Sciences and Technology of Saarland University, Saarbrücken, 2015.
76. *Bakali E., Chalki A., Kanellopoulos S., et al.* On the Power of Counting the Total Number of Computation Paths of NPTMs. `https://arxiv.org/abs/2306.11614`. 2024.

ПРИКЛАДНАЯ ТЕОРИЯ АВТОМАТОВ

УДК 519.711 DOI 10.17223/20710410/70/2

DECOMPOSITION OF A PARALLEL AUTOMATON INTO A NET OF SEQUENTIAL AUTOMATA

Yu. V. Pottosin

United Institute of Informatics Problems, National Academy of Sciences of Belarus, Minsk, Belarus

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

A method to construct a net of sequential automata that realizes the given parallel automaton is described. The parallelism relation of partial states is used to decompose a given parallel automaton. Each component automaton's set of states is based on mutually nonparallel partial states of the given parallel automaton. The state assignment of a component automaton provides decreasing power consumption of the designed device based on reducing the switching activity of memory elements. The joint low power assignment of states of component automata takes into consideration the conditional compatibility of states. The component automata exchange with binary signals. The communication between component automata is minimized.

Keywords: *parallel automaton, partial state, state assignment, complete bipartite sub-graph, weighted cover problem.*

ДЕКОМПОЗИЦИЯ ПАРАЛЛЕЛЬНОГО АВТОМАТА В СЕТЬ ПОСЛЕДОВАТЕЛЬНЫХ АВТОМАТОВ

Ю. В. Поттосин

Объединенный институт проблем информатики НАН Беларуси, г. Минск, Беларусь

Описан способ построения сети из последовательных автоматов, реализующей заданный параллельный автомат. При декомпозиции используется отношение параллельности частичных состояний заданного параллельного автомата. Множество состояний каждого из компонентных последовательных автоматов образуется на основе множества взаимно непараллельных частичных состояний заданного параллельного автомата. Кодирование состояний компонентного автомата предусматривает уменьшение энергопотребления проектируемого устройства на основе снижения интенсивности переключений элементов памяти. При совместном энергосберегающем кодировании состояний компонентных автоматов учитывается условная совместимость состояний. Компонентные автоматы обмениваются двоичными сигналами. Число межкомпонентных связей минимизируется.

Ключевые слова: *параллельный автомат, частичное состояние, декомпозиция автоматов, энергосберегающее кодирование состояний автомата, задача взвешенного покрытия.*

1. Introduction

A parallel automaton is a functional model of a discrete device that gives a concise representation of the parallelism of controlled interactive processes [1]. This model is close to widely known Petri net [2]. Unlike the classical finite sequential automaton, the parallel automaton can be in several states simultaneously. They are called *partial*. The set of all partial states that a parallel automaton can be in at some time, not being in any other state, is called *global state*.

The application of the decomposition way gives the possibility to lower the dimension of laborious tasks that appear in designing discrete devices. By decomposing a designed device into separate units, it becomes possible to decrease its power consumption by blocking clocks supplied to some units [3]. The decomposition of a parallel automaton into a net of sequential automata is worthwhile because the model of a sequential automaton allows using wide known effective methods for logical design of discrete devices. This paper considers, as well, such problems of logical design as state assignment and constructing Boolean functions of memory element excitation. The state assignment of component automata is fulfilled jointly. The problem of minimizing interconnections is considered as well.

The problem of state assignment takes a special place in logical design of discrete devices. Its solving influences considerably on complexity and power consumption of a designed device. The amount of power consumption is one of the main optimization criteria in designing discrete devices. It is caused by the tendency to increase the working time of power supply for portable devices and, on the other hand, by the tendency to lower acuity of the problem of heat rejection in designing VLSI circuits. As it is said in [4, 5], the power consumption of a circuit built on the base of CMOS technology is proportional to switching activity of its logical and memory elements. It allows solving this problem at the level of logical design. In particular, decreasing power consumption can be achieved at the stage of state assignment of an automaton.

2. The used model

A parallel automaton consists of the following objects: a set of partial states $Q = \{q_1, q_2, \ldots, q_\gamma\}$, a set of input Boolean variables $X = \{x_1, x_2, \ldots, x_n\}$, a set of output Boolean variables $Y = \{y_1, y_2, \ldots, y_m\}$ and a set of transitions $\{\tau_1, \tau_2, \ldots, \tau_t\}$, which is a sequence of lines of the following form [6]:

$$\tau_i = S_i : -K_i \to K_i' \to S_i', \tag{1}$$

where S_i and S_i' are subsets of Q, K_i is an elementary conjunction of variables from the set X, and K_i' is an elementary conjunction of variables from the set Y.

The sense of the line (1) is the following. If a partial automaton is in states forming the set S_i and Boolean variables took values that convert K_i to 1, then K_i' takes value 1 and the automaton comes to partial states in S_i' from the states composing S_i. In other words, let $P = \{P_1, P_2, \ldots, P_p\}$ be the set of all reachable global states of a given parallel automaton. Then if $S_i \subseteq P_g$, where P_g is a current global state of the automaton, and the automaton receives binary signals that turn the conjunction K_i into 1, then the global state will be $P_h = (P_g \setminus S_i) \cup S_i'$ at the next time and the automaton will produce binary signals that turn K_i' into 1. Any conjunction, K_i and K_i', can be absent in the line. The absence of K_i means its identical equality to 1. The absence of K_i' means, according to interpretation of the model, that either all the variables in Y are equal to 0 or the values of them do not change. In this paper, we do not consider output signals and omit K_i'. As well as for a sequential automaton, the synchronous or asynchronous implementation can be for

a parallel automaton. In a synchronous implementation, the time is divided into fixed units, during which the automaton goes from one state to another. In an asynchronous implementation, the duration of the unit is not fixed and is determined by the points of changing external input signals. When a signal comes to input, the automaton goes to stable partial states and do not go from them up to the next changing input. This changing must not occur until the partial states being stable.

The following restrictions are introduced in the model:

1) The initial global state is a one-element set. For the sake of determinacy, it can be $\{q_1\}$.
2) For two different lines, i-th and j-th, $S_i = S_j$ if $S_i \cap S_j \neq \varnothing$.

There are a number of other restrictions given in [1] and connected with correctness of an automaton description. They will not be considered here, as the correctness problem is not regarded here. Also, the output signals will not be considered here.

Example 1. Let a parallel automaton be defined by the following sequence of lines:

$$\tau_1 = 1 : - \overline{x}_1 x_2 \to 10; \qquad \tau_5 = 4 : - \overline{x}_1 \to 7;$$
$$\tau_2 = 10 : - \overline{x}_2 \to 2.3.4; \qquad \tau_6 = 7 : - \overline{x}_2 \to 9;$$
$$\tau_3 = 2 : - x_1 \to 5; \qquad \tau_7 = 8.9 : - \overline{x}_2 \to 6;$$
$$\tau_4 = 3.5 : - x_2 \to 8; \qquad \tau_8 = 6 : - x_1 \to 1.$$

Here $Q = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ and $X = \{x_1, x_2\}$. Let us take one-element set $\{1\}$ as an initial global state. The first line (transition τ_1) means that the automaton goes from state $\{1\}$ and comes to state $\{10\}$ in the next time unit if $x_1 = 0$ and $x_2 = 1$. The state $\{10\}$ is a global one, as well. The automaton stays in state $\{1\}$ at any other value combination of x_1 and x_2. The automaton goes from global state $\{10\}$ to partial states 2, 3 and 4 at $x_2 = 0$ (transition τ_2). Those partial states constitute the global state $\{2, 3, 4\}$. At the next time unit, the automaton changes the partial state 2 for partial state 5 (transition τ_3) and the automaton will be in global state $\{3, 4, 5\}$. Having observed the functioning of the automaton in this way, we get global states $\{2, 3, 7\}$, $\{2, 3, 9\}$, $\{3, 5, 7\}$, $\{3, 5, 9\}$, $\{7, 8\}$, $\{4, 8\}$, $\{8, 9\}$ and $\{6\}$.

3. Constructing a set of sequential automata implementing a given parallel automaton

A way of constructing a set of sequential automata implementing a given parallel automaton is described in [7, 8]. Let a parallel automaton B with a set Q of partial states be given. Its description is a sequence of lines of the form (1). Let us consider a set $N = (X, A_1, A_2, \ldots, A_n)$, where X is the set of input Boolean variables, the same as in the description of B, and $A_1, A_2, \ldots, A_n$ are component automata with sets $Q_1, Q_2, \ldots, Q_n$ of states. Each automaton A_j is one without outputs, i.e., only transitions are given for them in the form similar to (1):

$$q_j : - \alpha_j \to q^{j'}, \tag{2}$$

where α_j is a predicate of variables in X and states of component automata A_1, A_2, …, A_{j-1}, A_{j+1}, …, A_n. Its value is 1 at a certain value combination of some input variables and a certain set of states of some component automata. The automaton A_j goes to state $q^{j'}$ from q^j if $\alpha_j = 1$, otherwise it remains in q^j.

A net N *implements* an automaton B if there exists a mapping φ of $D \subseteq Q_1 \times Q_2 \times \ldots \times Q_n$ into a set of subsets of Q, such that for any transition (1) of B at $S_i \subseteq \varphi(q^1, q^2, \ldots, q^n)$

it goes to $S'_i \subseteq \varphi(q^{1'}, q^{2'}, \ldots, q^{n'})$ if $\alpha_j = 1$, and $S'_i \subseteq \varphi(q^1, q^2, \ldots, q^n)$ if $\alpha_j = 0$, according to (2). The described net N implements in the sense of [9, 10] a sequential automaton A, modelling B with global states of B.

The set of states Q_j of a component automaton A_j $(j = 1, 2, \ldots, n)$ of N is specified as follows. Every state $q^j \in Q_j$ is in a mutually one-to-one mapping with a partial state q of a parallel automaton B. Among the partial states that correspond to states in Q_j, no pair can be parallel. Partial states of a parallel automaton are called parallel if the automaton can be in them at the same time. The mapping is denoted as $f_j(q^j) = q$. Let the set $\{A_1, A_2, \ldots, A_n\}$ of component automata of N be enough to exist at least one component A_j with such a state q^j that $f_j(q^j) = q$.

The transitions in the component automata are specified according to (2) as follows. Each line τ_i of the form (1) from the description of B corresponds to the set of transitions in those component automata A_j, each of which has such a state q^j that $f_j(q^j) \in S_i \cup S'_i$. The left part of (2) for the component automaton A_j is q^j, for which $f_j(q^j) \in S_i$. There is exactly one such a state in Q_j because, according to determination of Q_j, no pair of states in Q_j corresponds to a pair of parallel partial states of B. The state $q^{j'}$ in (2) is determined such that $f_j(q^{j'}) \in S'_i$. At that, $\alpha_j = 1$ if and only if $K_i = 1$ and $\{f_1(q^1), f_2(q^2), \ldots, f_{j-1}(q^{j-1}), f_{j+1}(q^{j+1}), \ldots, f_n(q^n)\} \supseteq S'_i$.

The set of sequential automata $\{A_1, A_2, \ldots, A_n\}$, which constitutes the net N implementing the given parallel automaton B, is constructed in the following way. We obtain all maximal sets of mutually non-parallel partial states of B. The set is maximal in the sense that any partial state not belonging to it is parallel to some state belonging to it. Then, a set of these sets covering all the pairs of partial states connected in transitions must be obtained. Each set in the obtained cover corresponds to the set of states of one of the component sequential automaton constituting the desired net N.

In the case of low power assignment of a parallel automaton, the global states should be considered and the parallelism of partial states can be easily determined according to the global states. For the parallel automaton from the Example 1, the matrix of parallelism relation of partial states is as follows, where only the elements above the main diagonal are present because of symmetry of the relation:

$$
\begin{array}{c}
\begin{array}{ccccccccc} 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{array} \\
\left[\begin{array}{ccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
 & & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\
 & & & 1 & 0 & 0 & 1 & 0 & 0 \\
 & & & & 0 & 1 & 0 & 1 & 0 \\
 & & & & & 0 & 0 & 0 & 0 \\
 & & & & & & 1 & 0 & 0 \\
 & & & & & & & 1 & 0 \\
 & & & & & & & & 0
\end{array}\right]
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array}
\end{array} .
$$

We consider this matrix as the adjacency matrix of the graph representing parallelism relation. The mentioned sets of non-parallel partial states correspond to the independent sets in the graph. The methods to find independent sets in a graph are described in [11]. The sets $\{1, 2, 5, 6, 8, 10\}$, $\{1, 3, 6, 8, 10\}$ and $\{1, 4, 6, 7, 9, 10\}$ are independent sets in the graph under consideration. All the sets constitute the only shortest cover, and so, the cover problem must not be solved as it is done in [7, 8].

The states of the component automata are convenient to be denoted by the same symbols that the corresponding partial states of the given parallel automaton are denoted. So three

automata will constitute the desired net: A_1 with set of states $Q_1 = \{1, 2, 5, 6, 8, 10\}$, A_2 with set of states $Q_2 = \{1, 3, 6, 8, 10\}$ and A_3 with set of states $Q_3 = \{1, 4, 6, 7, 9, 10\}$. The transitions between states are determined as it is described above according to (2). We give the behavior of the obtained net N as a system of discrete functions $q^{1'}$, $q^{2'}$ and $q^{3'}$ that take values from Q_1, Q_2 and Q_3, respectively. Their arguments are Boolean variables x_1, x_2 and multivalued variables q^1, q^2 and q^3 that take values from Q_1, Q_2 and Q_3. Table 1, which has the generally accepted form of flow tables for an automaton, represents this system. The rows of Table 1 correspond to the states of the component automata, and the columns to the triplet $x_i, x_j, q^{k'}$ $(i, j = 1, 2;\ k = 1, 2, 3)$. Any element of Table 1 represents the state where the automaton goes under the conditions shown in the row and column.

Table 1

q^1	q^2	q^3	x_1x_2											
			00			01			11			10		
			$q^{1'}$	$q^{2'}$	$q^{3'}$	$q^{1'}$	$q^{2'}$	$q^{3'}$	$q^{1'}$	$q^{2'}$	$q^{3'}$	$q^{1'}$	$q^{2'}$	$q^{3'}$
1	1	1	1	1	1	10	10	10	1	1	1	1	1	1
10	10	10	2	3	4	10	10	10	10	10	10	2	3	4
2	3	4	2	3	7	2	3	7	5	3	4	5	3	4
2	3	7	2	3	9	2	3	7	5	3	7	5	3	9
5	3	4	5	3	7	8	8	7	8	8	4	5	3	4
2	3	9	2	3	9	2	3	9	5	3	9	5	3	9
5	3	7	5	3	9	8	8	7	8	8	7	5	3	9
5	3	9	5	3	9	8	8	9	8	8	9	5	3	9
8	8	7	8	8	9	8	8	7	8	8	7	8	8	9
8	8	4	8	8	7	8	8	7	8	8	4	8	8	4
8	8	9	6	6	6	8	8	9	8	8	9	6	6	6
6	6	6	6	6	6	6	6	6	1	1	1	1	1	1

As a result of state assignment, when any variable q^j (relatively, $q^{j'}$) is replaced by a Boolean vector with components z_i^j (relatively, $z_i^{j'}$) representing states of memory elements, the system of functions given by Table 1 is transformed into a system of Boolean functions.

4. The method for state assignment of a sequential automaton

The iterative way described in [11] for state assignment is used. The search for the maximum cut in a weighted graph is based on it. Let the state assignment of an automaton A be required. That is, the Boolean vectors $(z_1, z_2, \ldots, z_k)$, called state codes, must be assigned to the states q of A, and different states are assigned with different codes. Partial codes $(z_1, z_2, \ldots, z_j)$, $j < k$, and a weighted graph $G = (V, E)$, whose vertices correspond to the states of A, describe the current situation in the way fulfilling. An edge connects two vertices in G if and only if the corresponding states have the same partial code. The partial codes are empty and G is a complete graph in the initial situation. Every edge $v_sv_t \in E$ has a weight w_{st} proportional to $1 - p_{st}$, where p_{st} is the probability of transition between states q_s and q_t (independently of direction) corresponding to vertices v_s and v_t. Evidently, the probability p_{st} is equal to the sum of the probabilities of transitions from q_s to q_t and from q_t to q_s. To lower the switching activity of memory elements, the Hamming distance between the codes of states q_s and q_t in the space of Boolean variables $z_1, z_2, \ldots, z_k$ must be made shorter if p_{st} is high.

The process of state assignment of a given automaton is a sequence of steps. At the i-th step, a partition of the vertex set V of G into two subsets, V_1 and V_2, is obtained. The variable z_i is introduced and receives the value 0 (or 1) for the states corresponding

to vertices in V_1 and value 1 (or 0) for the states corresponding to vertices in V_2 if those vertices are ends of any edge. Then, the edges connecting vertices in V_1 with vertices in V_2 are removed, and the next step, $(i+1)$-th one, is fulfilled. The process is over when graph G becomes empty.

The problem to partition V into V_1 and V_2 is reduced to finding the maximum cut in G, i.e., finding such a partition that the sum of weights of the edges connecting the vertices of V_1 to the vertices of V_2 would be maximum. At the last step, only those edges remain that correspond to the pairs of states connected with transitions of comparatively large probabilities. The Hamming distances between the codes of those states are equal to one. To find the cut, the method described in [12] can be applied. It is a sequence of steps, at each of which a vertex v is selected in V_1 and carried to V_2. The initial meanings are $V_1 = \varnothing$ and $V_2 = V$, and the vertex v is selected in the following way.

Let d be the sum of weights of the edges incident with $v \in V_2$, and c be the sum of weights of edges connecting v with the vertices in V_1. The transfer of the vertex v from V_2 to V_1 increases the sum of weights of edges connecting the vertices from V_1 with the vertices in V_2 by $h = d-2c$ if it is positive. At the first step, h is equal to d. At any step, the vertex v with maximum value of h is selected. The process comes to the end when h is not positive for all the vertices in V_2.

5. Calculating probabilities of transitions

The following assumptions are accepted in calculating probabilities of transitions between states of a sequential automaton. The automaton is completely specified; all the states are mutually reachable, i.e., for every two states there exists a sequence of input signals transferring one of them to the other; the automaton works a long time enough.

The probability of transition of a sequential automaton from a state q_i to a state q_j caused by an input signal $x = (x_1, x_2, \ldots, x_n)$ is equal to the probability of coming x. If there are several input signals transferring the automaton from q_i to q_j, the conditional probability p'_{ij} of such a transfer is equal to the sum of the probabilities of those signals as a probability of incompatible events. The condition is that the automaton is in the state q_i. The absolute probability p_{ij} of the transition from q_i to q_j for all the time of the automaton working is equal to the product $p(q_i)p'_{ij}$, where $p(q_i)$ is the probability that the automaton is in the state q_i — this event and the incoming signals that transfer the automaton from q_i to q_j are independent events.

To calculate the probabilities $p(q_i)$, $i = 1, 2, \ldots, m$, where m is the number of states of the automaton, the Chapmann — Kolmogorov equations for discrete-time Markov Chains [13] can be applied. That method was used in [7, 8, 14]. Similarly to Kirchhoff's low in electrical engineering, one may say that the sum of the probabilities of transitions to some state is equal to the sum of the probabilities of transitions from this state. Based on the considerations above, the following equations with unknown quantities $p(q_i)$ $(i = 1, 2, \ldots, m)$ can be derived:

$$\sum_{i=1}^{m} p(q_i)p'_{ij} = p(q_j), \quad j = 1, \ldots, m,$$

$$\sum_{i=1}^{m} p(q_i) = 1.$$

The probabilities p'_{ij} must be known. So, having solved this system of equations, the probabilities $p(q_i)$ will be obtained. As it was said above, the absolute probability p_{ij} is defined as $p_{ij} = p(q_i)p'_{ij}$.

An input signal of a component automaton in a net consists of an input signal of the net and signals on the states of the other components of the net. Due to interconnection in the network, the state of a component automaton and the input signal it receives are not independent events. To calculate the probabilities of transitions between component automata, the modeling a parallel automaton by a sequential automaton is suggested. The states of the modeling sequential automaton are the global states of the given parallel automaton. The probabilities of transitions between the global states are defined as it is shown above for the modeling sequential automaton. The probabilities of transitions between the partial states and, relatively, between the states of component automata are determined using the probabilities of transitions between global states. The probability of the transition between partial states q_i and q_j is equal to the sum of the probabilities of the transitions between those global states P_s and P_t, for which $q_i \in P_s$ and $q_j \in P_t$, or $q_i \in P_t$ and $q_j \in P_s$.

The transitions between the global states of the parallel automaton in the considered example is shown in Table 2, where rows and columns correspond to global states, and its entry is the condition of the transition from the global state corresponding to the row into the global state corresponding to the column. Using Table 2, the conditional probabilities are determined easily. Table 3 shows them with common denominator. Here, we regard binary input signals x_1 and x_2 as independent and equally probable.

Table 2

States	$\{1\}$	$\{10\}$	$\{2,3,4\}$	$\{2,3,7\}$	$\{3,4,5\}$	$\{2,3,9\}$	$\{3,5,7\}$	$\{3,5,9\}$	$\{7,8\}$	$\{4,8\}$	$\{8,9\}$	$\{6\}$
$\{1\}$	$x_1 \vee \overline{x}_2$	$\overline{x}_1 x_2$										
$\{10\}$		x_2	$\overline{x}_2$									
$\{2,3,4\}$				$\overline{x}_1$	x_1							
$\{2,3,7\}$				$\overline{x}_1 x_2$		$\overline{x}_1 \overline{x}_2$	$x_1 x_2$	$x_1 \overline{x}_2$				
$\{3,4,5\}$					$x_1 \overline{x}_2$		$\overline{x}_1 \overline{x}_2$		$\overline{x}_1 x_2$	$x_1 x_2$		
$\{2,3,9\}$						$\overline{x}_1$		x_1				
$\{3,5,7\}$								$\overline{x}_2$	x_2			
$\{3,5,9\}$								$\overline{x}_2$			x_2	
$\{7,8\}$									x_2		$\overline{x}_2$	
$\{4,8\}$									$\overline{x}_1$	x_1		
$\{8,9\}$											x_2	$\overline{x}_2$
$\{6\}$	x_1											$\overline{x}_1$

Table 3

States	$\{1\}$	$\{10\}$	$\{2,3,4\}$	$\{2,3,7\}$	$\{3,4,5\}$	$\{2,3,9\}$	$\{3,5,7\}$	$\{3,5,9\}$	$\{7,8\}$	$\{4,8\}$	$\{8,9\}$	$\{6\}$
$\{1\}$	3/4	1/4										
$\{10\}$		2/4	2/4									
$\{2,3,4\}$				2/4	2/4							
$\{2,3,7\}$				1/4		1/4	1/4	1/4				
$\{3,4,5\}$					1/4		1/4		1/4	1/4		
$\{2,3,9\}$						2/4		2/4				
$\{3,5,7\}$								2/4	2/4			
$\{3,5,9\}$								2/4			2/4	
$\{7,8\}$									2/4		2/4	
$\{4,8\}$									2/4	2/4		
$\{8,9\}$											2/4	2/4
$\{6\}$	2/4											2/4

The following system of equations is obtained for the probabilities of global states $\{1\}$, $\{10\}$, $\{2,3,4\}$, $\{2,3,7\}$, $\{3,4,5\}$, $\{2,3,9\}$, $\{3,5,7\}$, $\{3,5,9\}$, $\{7,8\}$, $\{4,8\}$, $\{8,9\}$ and $\{6\}$ according to Table 3:

$$
\begin{aligned}
p(1) &= 3/4\,p(1) + 2/4\,p(6),\\
p(10) &= 2/4\,p(10) + 1/4\,p(1),\\
p(2.3.4) &= 2/4\,p(10),\\
p(2.3.7) &= 2/4\,p(2.3.4) + 1/4\,p(2.3.7),\\
p(3.4.5) &= 2/4\,p(2.3.4) + 1/4\,p(3.4.5),\\
p(2.3.9) &= 2/4\,p(2.3.9) + 1/4\,p(2.3.7),\\
p(3.5.7) &= 1/4\,p(3.4.5) + 1/4\,p(2.3.7),\\
p(3.5.9) &= 2/4\,p(2.3.9) + 1/4\,p(2.3.7) + 2/4\,p(3.5.7) + 2/4\,p(3.5.9),\\
p(7.8) &= 2/4\,p(7.8) + 1/4\,p(3.4.5) + 2/4\,p(3.5.7) + 2/4\,p(4.8),\\
p(4.8) &= 2/4\,p(4.8) + 1/4\,p(3.4.5),\\
p(8.9) &= 2/4\,p(7.8) + 2/4\,p(8.9) + 2/4\,p(3.5.9),\\
p(6) &= 2/4\,p(6) + 2/4\,p(8.9),\\
&p(1) + p(10) + p(2.3.4) + p(2.3.7) + p(3.4.5) + p(2.3.9) + p(3.5.7) + p(3.5.9) +\\
&\quad + p(7.8) + p(4.8) + p(8.9) + p(6) = 1.
\end{aligned}
$$

The solution of this system gives the probabilities

$$
\begin{aligned}
p(1) &= 12/46, & p(10) &= 6/46, & p(2.3.4) &= 3/46, & p(2.3.7) &= 2/46,\\
p(3.4.5) &= 2/46, & p(2.3.9) &= 1/46, & p(3.5.7) &= 1/46, & p(3.5.9) &= 3/46,\\
p(7.8) &= 3/46, & p(4.8) &= 1/46, & p(8.9) &= 6/46, & p(6) &= 6/46.
\end{aligned}
$$

The absolute probabilities of transitions between global states according to $p_{ij} = p(q_i)p'_{ij}$ are in Table 4.

Table 4

States	$\{1\}$	$\{10\}$	$\{2,3,4\}$	$\{2,3,7\}$	$\{3,4,5\}$	$\{2,3,9\}$	$\{3,5,7\}$	$\{3,5,9\}$	$\{7,8\}$	$\{4,8\}$	$\{8,9\}$	$\{6\}$
$\{1\}$	18/92	6/92										
$\{10\}$		6/92	6/92									
$\{2,3,4\}$				3/92	3/92							
$\{2,3,7\}$				1/92		1/92	1/92	1/92				
$\{3,4,5\}$					1/92		1/92		1/92	1/92		
$\{2,3,9\}$						1/92		1/92				
$\{3,5,7\}$								1/92	1/92			
$\{3,5,9\}$								3/92			3/92	
$\{7,8\}$									3/92		3/92	
$\{4,8\}$									1/92	1/92		
$\{8,9\}$											6/92	6/92
$\{6\}$	6/92											6/92

As it is said above, the probabilities of transitions between partial states are determined using the probabilities of transitions between global states. For example, the automaton in the considered example goes from partial state 2 to partial state 5 (transition τ_3) when it goes from global states $\{2,3,4\}$, $\{2,3,7\}$ and $\{2,3,9\}$ to global states $\{3,4,5\}$, $\{3,5,7\}$ and $\{3,5,9\}$. The sum of the probabilities of those incompatible events is 6/92. It is the probability of the transition from partial state 2 to partial state 5. The component automaton A_1 goes from state 2 to state 5 with the same probability.

6. Joint state assignment of component automata

As the weight of an edge in the graph G_i for the automaton A_i, $i = 1, 2, 3$, we take the numerator of fraction $1 - p^*_{st}$, where p^*_{st} is the probability of the transition in any direction between states q_s and q_t that correspond to the end of the edge. The adjacency matrices of weighted graphs G_1, G_2 and G_3 for automata A_1, A_2 and A_3, where the values below the main diagonal are omitted because of symmetry of the matrices, are the following:

$$A_1 = \begin{array}{c} \begin{matrix} 2 & 5 & 6 & 8 & 10 \end{matrix} \\ \left[\begin{array}{ccccc} 92 & 92 & 86 & 92 & 86 \\ & 86 & 92 & 92 & 86 \\ & & 92 & 86 & 92 \\ & & & 86 & 92 \\ & & & & 92 \end{array}\right] \end{array} \begin{matrix} 1 \\ 2 \\ 5 \\ 6 \\ 8 \end{matrix}, \quad A_2 = \begin{array}{c} \begin{matrix} 3 & 6 & 8 & 10 \end{matrix} \\ \left[\begin{array}{cccc} 92 & 86 & 92 & 86 \\ & 92 & 86 & 86 \\ & & 86 & 92 \\ & & & 92 \end{array}\right] \end{array} \begin{matrix} 1 \\ 3 \\ 6 \\ 8 \end{matrix},$$

$$A_3 = \begin{array}{c} \begin{matrix} 4 & 6 & 7 & 9 & 10 \end{matrix} \\ \left[\begin{array}{ccccc} 92 & 86 & 92 & 92 & 86 \\ & 92 & 86 & 92 & 86 \\ & & 92 & 86 & 92 \\ & & & 86 & 92 \\ & & & & 92 \end{array}\right] \end{array} \begin{matrix} 1 \\ 4 \\ 6 \\ 7 \\ 9 \end{matrix}.$$

The method described above is applied for the search for the maximum cuts in G_1, G_2 and G_3. The maximal values of d in G_1, G_2 and G_3 determine the result of the first step of the partition. They are $d_1(1) = 448$, $d_2(1) = 356$ and $d_3(1) = 448$. According to them, the following partitions are obtained: $\{\{1\}, \{2, 5, 6, 8, 10\}\}$, $\{\{1\}, \{3, 6, 8, 10\}\}$ and $\{\{1\}, \{4, 6, 7, 9, 10\}\}$. The next partitions are determined by the maximal values $h_1(6) = 276$, $h_2(6) = 184$ and $h_3(6) = 276$: $\{\{1, 6\}, \{2, 5, 8, 10\}\}$, $\{\{1, 6\}, \{3, 8, 10\}\}$ and $\{\{1, 6\}, \{4, 7, 9, 10\}\}$. Finally, according to $h_1(8) = 92$, $h_2(10) = 0$ and $h_3(9) = 92$, we obtain the cuts of the graphs in the form of partitions $\{\{1, 6, 8\}, \{2, 5, 10\}\}$, $\{\{1, 6\}, \{3, 8, 10\}\}$ and $\{\{1, 6, 9\}, \{4, 7, 10\}\}$.

The values of the coding variables z^1_i, z^2_i and z^3_i, which are introduced after having found the recurrent cuts, are determined taking into consideration the state compatibility. After i-th step in the process of state assignment, the variable z^j_i must take different values for the states of A_j corresponding to the vertices from different parts of the cut of G_j if these states are incompatible, and must not take different values if they are compatible. The state compatibility of component automata is defined as it is described in [15], in the following way. Let $f_j(q^j_l) = f_k(q^k_m) = q_s$ and $f_j(q^j_u) = f_k(q^k_v) = q_t$, where q^j_l and q^j_u are the states of the component automaton A_j, q^k_m and q^k_v are the states of A_k ($j \neq k$), q_s and q_t are partial states of the given parallel automaton B, f_j and f_k are mappings of the states of A_j and A_k into partial states of B. One of the pairs, (q^j_l, q^j_u) and (q^k_m, q^k_v), can be regarded as compatible, while the other is incompatible. The optimal choice of it requires special investigation is not considered in this paper.

The following matrices give the values of z^1_1, z^2_1 and z^3_1:

$$\begin{array}{cccccc|c} 1 & 2 & 5 & 6 & 8 & 10 & q^1 \\ 0 & 1 & 1 & 0 & 0 & 1 & z^1_1 \end{array}, \quad \begin{array}{ccccc|c} 1 & 3 & 6 & 8 & 10 & q^2 \\ 0 & 1 & 0 & 1 & - & z^2_1 \end{array}, \quad \begin{array}{cccccc|c} 1 & 4 & 6 & 7 & 9 & 10 & q^3 \\ 0 & 1 & 0 & 1 & 0 & - & z^3_1 \end{array}.$$

Here, the states 1 and 10 of the automaton A_1 are incompatible, and the variable z^1_1 takes different values for them, while the values of z^2_1 and z^3_1 remain indefinite for the states of A_2 and A_3 mapped into the same partial state of B.

The following matrices are obtained after removing the edges corresponding to the pairs of states with different values of coding variables and to the pairs of compatible states:

$$A_1 = \begin{array}{c} \begin{matrix} 2 & 5 & 6 & 8 & 10 \end{matrix} \\ \left[\begin{array}{ccccc} 0 & 0 & 86 & 92 & 0 \\ & 86 & 0 & 0 & 86 \\ & & 0 & 0 & 92 \\ & & & 86 & 0 \\ & & & & 0 \end{array}\right] \end{array} \begin{matrix} 1 \\ 2 \\ 5 \\ 6 \\ 8 \end{matrix}, \quad A_2 = \begin{array}{c} \begin{matrix} 3 & 6 & 8 & 10 \end{matrix} \\ \left[\begin{array}{cccc} 0 & 86 & 0 & 0 \\ & 0 & 86 & 86 \\ & & 0 & 0 \\ & & & 0 \end{array}\right] \end{array} \begin{matrix} 1 \\ 3 \\ 6 \\ 8 \end{matrix}, \quad A_3 = \begin{array}{c} \begin{matrix} 4 & 6 & 7 & 9 & 10 \end{matrix} \\ \left[\begin{array}{ccccc} 0 & 86 & 0 & 92 & 0 \\ & 0 & 86 & 0 & 86 \\ & & 0 & 86 & 0 \\ & & & 0 & 92 \\ & & & & 92 \end{array}\right] \end{array} \begin{matrix} 1 \\ 4 \\ 6 \\ 7 \\ 9 \end{matrix}.$$

The partitions $\{\{1,5\},\{2,6,8,10\}\}$, $\{\{1,3\},\{6,8,10\}\}$, $\{\{7,9,10\},\{1,4,6\}\}$ and, relatively, the values of z_2^1, z_2^2 and z_2^3 are obtained by the same way:

$$\begin{array}{c} \begin{matrix} 1 & 2 & 5 & 6 & 8 & 10 \end{matrix} \\ \left[\begin{array}{cccccc} 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & - & 0 \end{array}\right] \end{array} \begin{matrix} q^1 \\ z_1^1 \\ z_2^1 \end{matrix}, \quad \begin{array}{c} \begin{matrix} 1 & 3 & 6 & 8 & 10 \end{matrix} \\ \left[\begin{array}{ccccc} 0 & 1 & 0 & 1 & - \\ - & 0 & - & 1 & 1 \end{array}\right] \end{array} \begin{matrix} q^2 \\ z_1^2 \\ z_2^2 \end{matrix}, \quad \begin{array}{c} \begin{matrix} 1 & 4 & 6 & 7 & 9 & 10 \end{matrix} \\ \left[\begin{array}{cccccc} 0 & 1 & 0 & 1 & 0 & - \\ 0 & 0 & 0 & 1 & 1 & 1 \end{array}\right] \end{array} \begin{matrix} q^3 \\ z_1^3 \\ z_2^3 \end{matrix}.$$

The values of z_2^1 are given by vector $(0\,0\,1\,1\,-\,0)$ according to the partition $\{\{1,5\}, \{2,6,8,10\}\}$ and taking into consideration the state compatibility. Vector $(1\,0\,1\,0\,-0)$ could give the values of z_2^1 as well, but then, the probability of the transitions between states with changing values of z_2^1 is 6/92, while there are no such transitions at the first variant. Therefore, the first variant gives less switching activity.

After removing edges, the graph G_2 has become empty and state assignment of A_2 has fulfilled. The matrices for A_1 and A_3 are

$$A_1 = \begin{array}{c} \begin{matrix} 2 & 5 & 6 & 8 & 10 \end{matrix} \\ \left[\begin{array}{ccccc} 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 86 \\ & & 0 & 0 & 0 \\ & & & 0 & 0 \\ & & & & 0 \end{array}\right] \end{array} \begin{matrix} 1 \\ 2 \\ 5 \\ 6 \\ 8 \end{matrix}, \quad A_3 = \begin{array}{c} \begin{matrix} 4 & 6 & 7 & 9 & 10 \end{matrix} \\ \left[\begin{array}{ccccc} 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 \\ & & 0 & 0 & 0 \\ & & & 0 & 92 \\ & & & & 92 \end{array}\right] \end{array} \begin{matrix} 1 \\ 4 \\ 6 \\ 7 \\ 9 \end{matrix}.$$

They make us to introduce variables z_3^1 and z_3^3. The following matrices represent the result of state assignment of the component automata:

$$\begin{array}{c} \begin{matrix} 1 & 2 & 5 & 6 & 8 & 10 \end{matrix} \\ \left[\begin{array}{cccccc} 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & - & 0 \\ - & 0 & - & - & - & 1 \end{array}\right] \end{array} \begin{matrix} q^1 \\ z_1^1 \\ z_2^1 \\ z_3^1 \end{matrix}, \quad \begin{array}{c} \begin{matrix} 1 & 3 & 6 & 8 & 10 \end{matrix} \\ \left[\begin{array}{ccccc} 0 & 1 & 0 & 1 & - \\ - & 0 & - & 1 & 1 \end{array}\right] \end{array} \begin{matrix} q^2 \\ z_1^2 \\ z_2^2 \end{matrix}, \quad \begin{array}{c} \begin{matrix} 1 & 4 & 6 & 7 & 9 & 10 \end{matrix} \\ \left[\begin{array}{cccccc} 0 & 1 & 0 & 1 & 0 & - \\ 0 & 0 & 0 & 1 & 1 & 1 \\ - & - & - & - & 0 & 1 \end{array}\right] \end{array} \begin{matrix} q^3 \\ z_1^3 \\ z_2^3 \\ z_3^3 \end{matrix}.$$

Substituting the vectors of values of the components z_i^j to the symbols of states in Table 1, we obtain the interval representation of the system of incompletely specified Boolean functions $z_i^{j'}$ of excitation of memory elements. It is convenient to represent the system in Table 5.

Table 5

x_1x_2	$z_1^1z_2^1z_3^1$	$z_1^2z_2^2$	$z_1^3z_2^3z_3^3$	$z_1^{1'}z_2^{1'}z_3^{1'}$	$z_1^{2'}z_2^{2'}$	$z_1^{3'}z_2^{3'}z_3^{3'}$
0 1	0 0 –	0 –	0 0 –	1 0 1	– 1	0 1 1
– 0	0 0 –	0 –	0 0 –	0 0 –	0 –	0 0 –
1 –	0 0 –	0 –	0 0 –	0 0 –	0 –	0 0 –
– 1	1 0 1	– 1	0 1 1	1 0 1	– 1	0 1 1
– 0	1 0 1	– 1	0 1 1	1 0 0	1 0	1 0 –
1 –	1 0 0	1 0	1 0 –	1 1 –	1 0	1 0 –
0 –	1 0 0	1 0	1 0 –	1 0 0	1 0	1 1 –
0 0	1 0 0	1 0	1 1 –	1 0 0	1 0	0 1 0
0 1	1 0 0	1 0	1 1 –	1 0 0	1 0	1 1 –
1 1	1 0 0	1 0	1 1 –	1 1 –	1 0	1 1 –
1 0	1 0 0	1 0	1 1 –	1 1 –	1 0	0 1 0
1 0	1 1 –	1 0	1 0 –	1 1 –	1 0	1 0 –
0 0	1 1 –	1 0	1 0 –	1 1 –	1 0	1 1 –
0 1	1 1 –	1 0	1 0 –	0 – –	1 1	1 1 –
1 1	1 1 –	1 0	1 0 –	0 – –	1 1	1 0 –
0 –	1 0 0	1 0	0 1 0	1 0 0	1 0	0 1 0
1 –	1 0 0	1 0	0 1 0	1 1 –	1 0	0 1 0
– 0	1 1 –	1 0	1 1 –	1 1 –	1 0	0 1 0
– 1	1 1 –	1 0	1 1 –	0 – –	1 1	1 1 –
– 0	1 1 –	1 0	0 1 0	1 1 –	1 0	0 1 0
– 1	1 1 –	1 0	0 1 0	0 – –	1 1	0 1 0
– 1	0 – –	1 1	1 1 –	0 – –	1 1	1 1 –
– 0	0 – –	1 1	1 1 –	0 – –	1 1	0 1 0
1 –	0 – –	1 1	1 0 –	0 – –	1 1	1 0 –
0 –	0 – –	1 1	1 0 –	0 – –	1 1	1 1 –
– 1	0 – –	1 1	0 1 0	0 – –	1 1	0 1 0
– 0	0 – –	1 1	0 1 0	0 1 –	0 –	0 0 –
0 –	0 1 –	0 –	0 0 –	0 1 –	0 –	0 0 –
1 –	0 1 –	0 –	0 0 –	0 0 –	0 –	0 0 –

7. Reduction of interconnection

The component sequential automata in the net implementing the given parallel automaton exchange with binary signals as the Boolean variables z_i^j. The problem of reducing these connections makes sense when the reduction of the number of input pins of a logic circuit is necessary. Solving this problem, as it is described in [16], by decreasing the number of arguments of Boolean excitation functions of memory elements for each component separately is suggested.

Let a pair of matrices $(\mathbf{X}, \mathbf{Y})$ be an interval representation of a system of incompletely specified Boolean functions, where the rows of $\mathbf{X}$ represent the intervals of Boolean space of the arguments, and the rows of $\mathbf{Y}$ the values of the functions at the corresponding intervals. The condition of correctness requires the row orthogonality of $\mathbf{X}$ and the orthogonality of the corresponding rows of $\mathbf{Y}$. The *distinction matrix* of the rows of $\mathbf{X}$ is constructed. For each pair of rows of $\mathbf{X}$ corresponding to orthogonal rows of $\mathbf{Y}$, the distinction matrix has a row obtained by modulo 2 addition of those rows of $\mathbf{X}$. The modulo 2 addition of don't care "–" with zero or one gives zero. The shortest column cover must be found in the distinction matrix, that is the least set of columns such that each row of the matrix has one in these columns. The obtained cover shows the set of essential arguments, and when selecting the columns to form the cover, we prefer those that correspond to the variables coding the states of the automaton under consideration. Thus, we decrease interconnection of components.

For the system of excitation functions in automata A_1, A_2 and A_3, the distinction matrices are as follows after applying the reduction rule in solving the cover problem (if row i has 1s everywhere, where row j has 1s, row i can be removed [17]):

$$\begin{array}{c} x_1\ x_2\ z_1^1\ z_2^1\ z_3^1\ z_1^2\ z_2^2\ z_1^3\ z_2^3\ z_3^3 \\ \left[\begin{array}{cccccccccc} 0&0&1&0&0&0&0&1&0&0\\ 0&0&1&0&0&0&0&0&1&0\\ 0&0&0&0&0&1&0&0&1&0\\ 0&0&0&1&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0&0&0\\ 1&0&0&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&1&0&0\\ 0&0&1&0&0&0&0&0&0&1\\ 0&0&0&0&1&0&1&1&0&0\\ 0&0&0&0&1&0&1&0&0&1 \end{array}\right] \end{array}, \quad \begin{array}{c} x_2\ z_1^1\ z_2^1\ z_3^1\ z_1^2\ z_2^2\ z_1^3\ z_2^3\ z_3^3 \\ \left[\begin{array}{ccccccccc} 0&0&0&0&0&0&1&0&0\\ 0&0&0&0&1&0&0&1&0\\ 0&1&0&0&0&0&0&1&0\\ 1&0&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0&1\\ 0&1&0&0&0&1&0&0&0\\ 0&0&0&1&0&1&0&0&1\\ 0&0&1&0&0&0&0&0&0 \end{array}\right] \end{array},$$

$$\begin{array}{c} x_1\ x_2\ z_1^1\ z_2^1\ z_3^1\ z_2^2\ z_1^3\ z_2^3\ z_3^3 \\ \left[\begin{array}{ccccccccc} 0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&1&0&0\\ 0&1&0&0&0&0&0&0&0\\ 1&0&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0&1\\ 0&0&1&0&0&1&0&0&0\\ 0&0&0&1&0&0&0&0&0\\ 0&0&0&0&1&1&0&0&1 \end{array}\right] \end{array}.$$

The process of forming the desired covers begins by introducing into them the columns z_1^1, z_2^1, z_3^1 from the first matrix, columns z_1^2, z_2^2 from the second matrix and z_1^3, z_2^3, z_3^3 from the third matrix. Relatively, the column covers $\{x_1, x_2, z_1^1, z_2^1, z_3^1, z_1^2\}$, $\{x_2, z_1^1, z_2^1, z_1^2, z_2^2, z_1^3\}$ and $\{x_1, x_2, z_1^1, z_2^1, z_1^3, z_2^3, z_3^3\}$ are obtained.

Figure 1 shows the net implementing the given parallel automaton. The following matrices represent the systems of disjunctive normal forms of the excitation functions in the component automata that are obtained separately for each automaton by the minimization method described in [18]:

$$\begin{array}{c} x_1\ x_2\ z_1^1\ z_2^1\ z_3^1\ z_1^2 \\ \left[\begin{array}{cccccc} 0&-&0&1&-&-\\ 0&1&-&0&-&0\\ -&0&0&-&-&1\\ -&1&-&-&1&-\\ -&-&1&0&-&-\\ 1&-&-&-&0&1\\ -&0&1&1&-&- \end{array}\right] \end{array}, \quad \begin{array}{c} z_1^{1'}\ z_2^{1'}\ z_3^{1'} \\ \left[\begin{array}{ccc} 0&1&0\\ 1&0&1\\ 0&1&0\\ 0&0&1\\ 1&0&0\\ 0&1&0\\ 1&1&0 \end{array}\right] \end{array}; \quad \begin{array}{c} x_2\ z_1^1\ z_2^1\ z_1^2\ z_2^2\ z_1^3 \\ \left[\begin{array}{cccccc} -&1&-&-&-&-\\ 1&-&-&-&1&-\\ -&0&-&-&-&-\\ 1&-&1&-&-&-\\ -&-&-&-&-&1\\ 1&-&-&1&-&- \end{array}\right] \end{array}, \quad \begin{array}{c} z_1^{2'}\ z_2^{2'} \\ \left[\begin{array}{cc} 1&0\\ 0&1\\ 0&1\\ 0&1\\ 1&0\\ 1&0 \end{array}\right] \end{array};$$

$$\begin{array}{c} x_1\ x_2\ z_1^1\ z_2^1\ z_1^3\ z_2^3\ z_3^3 \\ \left[\begin{array}{ccccccc} 0&-&-&0&-&0&-\\ -&1&1&-&-&1&1\\ -&-&-&-&1&1&-\\ -&0&1&-&-&-&1\\ 0&-&-&-&1&0&-\\ -&1&-&-&-&1&-\\ -&0&1&-&-&1&0\\ -&1&-&-&1&-&-\\ -&-&-&-&1&0&- \end{array}\right] \end{array}, \quad \begin{array}{c} z_1^{1'}\ z_2^{1'}\ z_3^{1'} \\ \left[\begin{array}{ccc} 0&1&1\\ 0&1&1\\ 0&1&0\\ 1&0&0\\ 0&1&0\\ 0&1&0\\ 0&1&0\\ 1&0&0\\ 1&0&0 \end{array}\right] \end{array}.$$

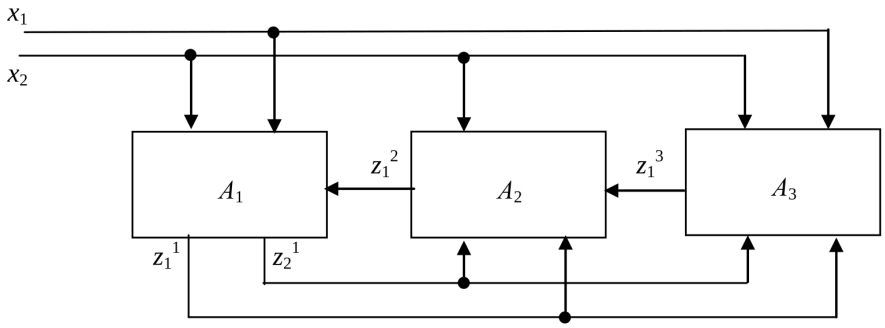

Fig. 1. Net of sequential automata

8. Asynchronous implementation

The probabilities of transitions between partial states of a parallel automaton were determined above with the help of modelling its behavior by a sequential automaton with the states corresponding to the global states of the parallel automaton. Not any parallel automaton admits such modelling under asynchronous implementation [1]. Here, we restrict ourselves to consideration of parallel automata that admit modelling by sequential automaton. The parallel automaton in the Example 1 is an example of such an automaton. We consider only transitions between stable states (partial and global) and regard the probability of unstable state to be negligibly small. The stable global states of the automaton in the Example 1 are $\{1\}$, $\{10\}$, $\{3,4,5\}$, $\{2,3,9\}$, $\{3,5,9\}$, $\{7,8\}$, $\{4,8\}$, $\{8,9\}$ and $\{6\}$.

The component sequential automata are determined by the same way that was applied above. The given parallel automaton is decomposed into asynchronous automata A_1, A_2 and A_3 with the sets of states $Q_1 = \{1,2,5,6,8,10\}$, $Q_2 = \{1,3,6,8,10\}$ and $Q_3 = \{1,4,6,7,9,10\}$. Table 6 shows transitions between states of the component automata, where the stable states are marked with bold.

Table 6

q^1	q^2	q^3	x_1x_2											
			00			01			11			10		
			$q^{1'}$	$q^{2'}$	$q^{3'}$	$q^{1'}$	$q^{2'}$	$q^{3'}$	$q^{1'}$	$q^{2'}$	$q^{3'}$	$q^{1'}$	$q^{2'}$	$q^{3'}$
1	1	1	**1**	**1**	**1**	10	10	10	**1**	**1**	**1**	**1**	**1**	**1**
10	10	10	2	3	9	**10**	**10**	**10**	**10**	**10**	**10**	5	3	4
5	3	4	5	3	9	8	8	7	8	8	4	**5**	**3**	**4**
2	3	9	**2**	**3**	**9**	**2**	**3**	**9**	8	8	9	5	3	9
5	3	9	**5**	**3**	**9**	8	8	9	8	8	9	**5**	**3**	**9**
8	8	7	6	6	6	**8**	**8**	**7**	**8**	**8**	**7**	1	1	1
8	8	4	6	6	6	8	8	7	**8**	**8**	**4**	**8**	**8**	**4**
8	8	9	6	6	6	**8**	**8**	**9**	**8**	**8**	**9**	1	1	1
6	6	6	**6**	**6**	**6**	**6**	**6**	**6**	1	1	1	1	1	1

For low power race-free state assignment of component automata, the approach described in detail in [19] is applied. The pairs of transitions between states at the same input signal are considered. For example, Table 6 shows that when $x_1 = 0$, $x_2 = 1$, the automata A_1, A_2 and A_3 have the pairs of transitions, $(1 \to 10, 5 \to 8)$, $(1 \to 10, 3 \to 8)$ and $(1 \to 10, 4 \to 7)$, relatively. The condition for absence of critical races in a pair of transitions is formulated by a ternary vector whose components correspond to the states of the automaton and have values 1 or 0 depending on what the transition of the pair the corresponding states belong to. For the pairs named above, (0−1−10), (01−10) and

(01−1−0) are such vectors where 0s and 1s can change places. Only one of such conditions is sufficient to fulfill. We call it obligatory. The optimal choice of obligatory conditions requires a special research.

All the obligatory conditions in the form of the vectors above constitute the condition matrix that has not implied rows [6]. A ternary vector **a** implies a ternary vector **b** if **b** is obtained from **a** by replacing some 0s or 1s by the value "−" and, perhaps, by inverting the obtained result. For example, vector (10−101) implies (10−−01) and (01−−1−). The sense of this relation is that the condition represented by **b** is satisfied if the condition represented by **a** is satisfied. For component automata of the considered net, the condition matrices can be as follows:

$$\begin{array}{cccccc} 1 & 2 & 5 & 6 & 8 & 10 \\ \left[\begin{array}{cccccc} - & 0 & 1 & - & - & 0 \\ 0 & - & - & 1 & 1 & - \\ 0 & 1 & 1 & - & 0 & - \\ 0 & - & 1 & - & 1 & 0 \\ - & 0 & - & - & - & 1 \end{array}\right] & \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \end{array},\quad \begin{array}{ccccc} 1 & 3 & 6 & 8 & 10 \\ \left[\begin{array}{ccccc} - & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & - & 1 \\ 0 & 1 & 0 & 1 & - \\ 0 & - & 1 & - & 0 \end{array}\right] & \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} \end{array},\quad \begin{array}{cccccc} 1 & 4 & 6 & 7 & 9 & 10 \\ \left[\begin{array}{cccccc} 0 & 1 & - & 0 & - & 1 \\ 0 & 1 & - & - & 0 & 1 \\ - & 0 & - & 0 & 1 & - \\ - & - & 0 & 0 & 1 & - \\ 0 & - & 0 & - & - & 1 \\ 0 & 1 & - & 0 & - & 1 \\ - & 1 & - & - & - & 0 \end{array}\right] & \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \end{array}.$$

A ternary matrix **R** implies a matrix **S** if for every row of **S** there is a row in **R** that implies it. The problem of race-free state assignment is reduced to finding a matrix with the minimal number of rows that implies the condition matrix and is called *shortest implying form* of the condition matrix. The rows of this matrix represent the desired codes of the states.

The shortest implying form of a ternary matrix is found in the following way. A set of rows of a matrix is called *compatible* if there is a vector implying each row of this set. A compatible set is the *maximal* one if it is not a proper subset of any other compatible set. We should find all the maximal compatible sets of the rows of the condition matrix and then obtain the shortest cover of the rows by these sets. Every compatible set correspond to the vector implying all the rows belonging to this set. The vectors corresponding to the elements of the obtained cover constitute the shortest implying form of the condition matrix under consideration. Below, the matrices are given whose rows imply the elements of the maximal compatible sets from the obtained shortest cover; the numbers of the rows of the condition matrix implied by each row of the given matrices are to their right:

$$\begin{array}{cccccc} 1 & 2 & 5 & 6 & 8 & 10 \\ \left[\begin{array}{cccccc} 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & - & 0 & 0 \end{array}\right] & \begin{array}{c} 1,2,4 \\ 3,5 \end{array} \end{array},\quad \begin{array}{ccccc} 1 & 3 & 6 & 8 & 10 \\ \left[\begin{array}{ccccc} 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{array}\right] & \begin{array}{c} 1,2 \\ 1,4 \\ 2,3 \end{array} \end{array},\quad \begin{array}{cccccc} 1 & 4 & 6 & 7 & 9 & 10 \\ \left[\begin{array}{cccccc} 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{array}\right] & \begin{array}{c} 1,2,5,6 \\ 1,4,5,6 \\ 2,3,4 \\ 3,4,5,7 \end{array} \end{array}.$$

We determine the relation for each variable z_i and the set of transitions between states with codes, where z_i changes its value at those transitions. That is the i-th memory element in the real circuit implementing the given automaton changes its state. The switching activity is connected with the probabilities of transitions between states. The probability of transition when z_i changes its value is put into the correspondence to z_i. This probability is equal to the sum of the probabilities of all the transitions when z_i changes its value, because those transitions are incompatible events.

The probabilities of the transitions (independently of the direction) between the partial states of the given parallel automaton coinciding with the probabilities of the transitions in component automata are obtained by the method described above. Table 7 shows them,

where the rows and columns correspond to the states and empty entries mean that there are no corresponding transitions.

Table 7

States	2	3	4	5	6	7	8	9	10
1					13/282	3/282	8/282	8/282	24/282
2				6/282			6/282		12/282
3						4/282	24/282		24/282
4						6/282		4/282	12/282
5							18/282		12/282
6						3/282	13/282	8/282	
7									
8									
9									12/282

Every compatible set of rows of the condition matrix and, correspondingly, vector implying all the rows in the set have the weight as the value proportional to the sum of the probabilities of the transitions connected with this vector. The minimal weight cover of the row sets of the condition matrix with the maximal compatible sets is obtained. The weight of a cover is the sum of the weights of its elements. The problem of minimal weighted cover is investigated in detail in [20].

All the family of maximal compatible sets for A_1 coincides with its shortest cover. According to the matrices above, the sets $\{1,4\}$ and $\{2,3\}$ constitute the shortest cover with the minimal weight for A_2, and $\{1,2,5,6\}$ and $\{3,4,5,7\}$ for A_3. The following matrices give the codes of the states of A_1, A_2 and A_3, respectively:

$$\begin{array}{cccccc} 1 & 2 & 5 & 6 & 8 & 10 \\ \left[\begin{array}{cccccc} 0 & 0 & 1 & 1 & 1 & - \\ 0 & 1 & 1 & - & 0 & 0 \end{array}\right] & \begin{array}{c} q^1 \\ z_1^1 \\ z_2^1 \end{array} \end{array}, \quad \begin{array}{ccccc} 1 & 3 & 6 & 8 & 10 \\ \left[\begin{array}{ccccc} 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{array}\right] & \begin{array}{c} q^2 \\ z_1^2 \\ z_2^2 \end{array} \end{array}, \quad \begin{array}{cccccc} 1 & 4 & 6 & 7 & 9 & 10 \\ \left[\begin{array}{cccccc} 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{array}\right] & \begin{array}{c} q^3 \\ z_1^3 \\ z_2^3 \end{array} \end{array}.$$

Table 8 is the interval specification of the system of incompletely specified Boolean functions $z_i^{j'}$ of excitation of memory elements. In this specification, the intervals of the space of internal variables that are determined by the transitions between states of the component automata are used [6].

After decreasing the number of arguments and minimization of the system of the excitation functions, the following matrices representing the systems of disjunctive normal forms are obtained:

$$\begin{array}{ccccccc} x_1 & x_2 & z_1^1 & z_2^1 & z_1^2 & z_2^2 & z_1^3 \\ \left[\begin{array}{ccccccc} - & 1 & 1 & - & - & 1 & - \\ 0 & - & 0 & 1 & - & - & - \\ 1 & 1 & - & - & - & 1 & 0 \\ - & - & 1 & 0 & - & - & 1 \\ 0 & - & 1 & - & 1 & - & - \\ 0 & 0 & - & - & - & 1 & - \\ 1 & 0 & - & - & 0 & - & 1 \\ - & 0 & 1 & 1 & - & 1 & - \\ 1 & 0 & - & 1 & - & 1 & - \end{array}\right] & , & \begin{array}{cc} z_1^{1'} & z_2^{1'} \\ \left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{array}\right] & ; \end{array} & \begin{array}{ccccccc} x_1 & x_2 & z_1^1 & z_2^1 & z_1^2 & z_2^2 & z_1^3 \\ \left[\begin{array}{ccccccc} 0 & - & - & - & 1 & - & - \\ 0 & - & - & - & 0 & 1 & - \\ 1 & - & - & - & - & - & 1 \\ 0 & 1 & 0 & - & - & - & - \\ - & 1 & 1 & - & - & 1 & - \\ 1 & 1 & - & - & - & 1 & 0 \\ - & - & - & - & 1 & - & 1 \\ 1 & - & 0 & 1 & - & 1 & - \\ - & 0 & - & 1 & - & 1 & - \end{array}\right] & , & \begin{array}{cc} z_1^{2'} & z_2^{2'} \\ \left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{array}\right] & ; \end{array} \end{array}$$

$$\begin{array}{c} \begin{matrix} x_1 & x_2 & z_1^1 & z_2^1 & z_1^2 & z_1^3 & z_2^3 \end{matrix} \\ \begin{bmatrix} 1 & - & - & - & - & 1 & - \\ - & 1 & 0 & - & - & 1 & - \\ 0 & 1 & 0 & 0 & - & - & - \\ - & 0 & - & 1 & 0 & 0 & - \\ - & 1 & 1 & - & - & - & 1 \\ 1 & 1 & - & - & - & - & 1 \\ 0 & - & - & - & - & - & 1 \\ 0 & 0 & - & 1 & 0 & - & - \end{bmatrix} \end{array}, \quad \begin{array}{c} \begin{matrix} z_1^{3'} & z_2^{3'} \end{matrix} \\ \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \end{array}.$$

Table 8

x_1x_2	$z_1^1z_2^1$	$z_1^2z_2^2$	$z_1^3z_2^3$	$z_1^{1'}z_2^{1'}$	$z_1^{2'}z_2^{2'}$	$z_1^{3'}z_2^{3'}$
− 0	0 0	0 0	0 0	0 0	0 0	0 0
1 −	0 0	0 0	0 0	0 0	0 0	0 0
0 1	0 0	0 −	0 −	0 0	0 1	1 1
− 1	0 0	0 1	1 1	0 0	0 1	1 1
0 0	0 −	0 1	− 1	0 1	0 1	0 1
1 0	− −	0 1	1 −	1 1	0 1	1 0
1 0	1 1	0 1	1 0	1 1	0 1	1 0
0 0	1 1	0 1	− −	1 1	0 1	0 1
0 1	1 −	− 1	− 0	1 0	1 1	0 0
1 1	1 −	− 1	1 0	1 0	1 1	1 0
0 −	0 1	0 1	0 1	0 1	0 1	0 1
1 1	− −	− 1	0 1	1 0	1 1	0 1
1 0	− 1	0 1	0 1	1 1	0 1	0 1
− 0	1 1	0 1	0 1	1 1	0 1	0 1
− 1	1 −	− 1	0 1	1 0	1 1	0 1
− 1	1 0	1 1	0 0	1 0	1 1	0 0
0 0	1 0	1 −	0 0	1 −	1 0	0 0
1 0	− 0	− −	0 0	0 0	0 0	0 0
1 −	1 0	1 1	1 0	1 0	1 1	1 0
0 1	1 0	1 1	− 0	1 0	1 1	0 0
0 0	1 0	1 −	− 0	1 −	1 0	0 0
− 1	1 0	1 1	0 1	1 0	1 1	0 1
1 0	− 0	− −	0 −	0 0	0 0	0 0
0 0	1 0	1 −	0 −	1 −	1 0	0 0
0 −	1 −	1 0	0 0	1 −	1 0	0 0
1 −	− −	1 0	0 0	0 0	− 0	0 0

Figure 2 shows the net of asynchronous sequential automata implementing the given parallel automaton.

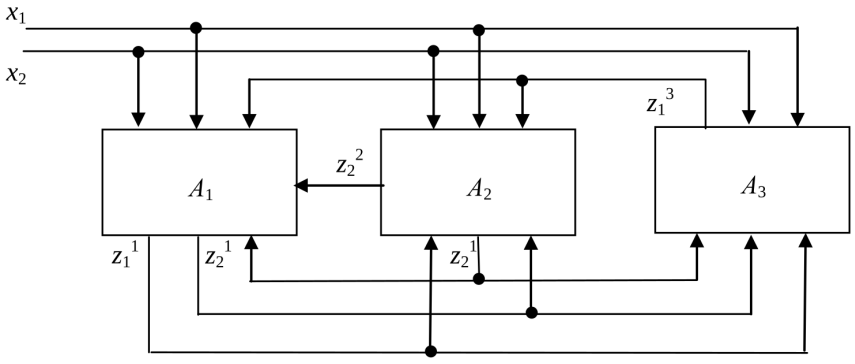

Fig. 2. Net of asynchronous sequential automata

9. Conclusion

The suggested approach to decomposition of a parallel automaton can be applied in the synthesis of distributed control systems. In such a system controlling a set of objects distant from each other, all the blocks are connected in a network, with each block located at one of the controlled objects. So the problem of minimization of interconnection considered in the paper is interesting from the point of view of reliability. Using the decomposition of a parallel automaton allows decreasing the dimension of laborious problems of logical design. The suggested approach is intended for using in a computer aided logical design system.

REFERENCES

1. *Zakrevskiy A. D.* Parallel'nye Algoritmy Logicheskogo Upravleniya [Parallel Algorithms for Logical Control]. Moscow, URSS, 2003, 304 p. (in Russian)
2. *Peterson J. L.* Petri Net Theory and the Modeling of Systems. Englewood Cliffs, N.J., Prentice-Hall Inc., 1981.
3. *Piguet C.* Low-power and low-voltage CMOS digital design. Microelectronic Eng., 1997, vol. 39, no. 1–4, pp. 179–208.
4. *Muroga S.* VLSI System Design. When and How to Design Very-Large-Scale Integrated Circuits. N.Y., John Wiley & Sons, 1982.
5. *Pedram M.* Power minimization in IC design: Principles and applications. ACM Trans. Design Automat. Electron. Syst., 1996, vol. 6, pp. 3–56.
6. *Zakrevskij A., Pottosin Yu., and Cheremisinova L.* Design of Logical Control Devices. Tallinn, TUT Press, 2009.
7. *Pottosin Yu. V.* Sovmestnoe energosberegayushchee kodirovanie sostoyaniy posledovatel'nykh avtomatov seti, realizuyushchey parallel'nyy avtomat [Joint low power state assignment of sequential automata of a net implementing a parallel automaton]. Informatika, 2023, vol. 20, no. 1, pp. 75–90. (in Russian)
8. *Pottosin Yu. V.* Dekompozitsiya parallel'nogo avtomata v set' posledovatel'nykh avtomatov i energosberegayushchee kodirovanie ikh sostoyaniy pri asinkhronnoy realizatsii [Decomposition of a parallel automaton into a net of sequential automata and low power state assignment of them at asynchronous implementation]. Informatika, 2024, vol. 21, no. 3, pp. 7–22. (in Russian)
9. *Hartmanis J. and Stearns R. E.* Algebraic Structure Theory of Sequential Machines. Englewood Cliffs, N.J., Prentice Hall Inc., 1966.
10. *Keevallik A. E.* Teorema dekompozitsii konechnykh avtomatov [A theorem of decomposition of finite automata]. Avtomatika i Vychislitel'naya Tekhnika, 1974, no. 1, pp. 17–24. (in Russian)
11. *Pottosin Yu. V.* Kombinatornye Zadachi v Logicheskom Proektirovanii Diskretnykh Ustroystv [Combinatorial Problems in Logical Design of Discrete Devices]. Minsk, Belaruskaja Navuka, 2021, 175 p. (in Russian)
12. *Zakrevskiy A. D.* Raskraska grafov pri dekompozitsii bulevykh funktsiy [Coloring graphs in decomposition of Boolean functions]. Logicheskoe Proektirovanie, 2000, iss. 5, pp. 83–97. (in Russian)
13. *Macii E., Pedram M., and Somenzi F.* High-level power modeling, estimation and optimization. IEEE Trans. CADICS, 1998, vol. 17, no. 11, pp. 1061–1079.
14. *Pottosin Yu. V.* Low power assignment of partial states of a parallel automaton. Prikladnaya Diskretnaya Matematika, 2022, no. 56, pp. 113–122.
15. *Pottosin Yu. V. and Shestakov E. A.* Dekompozitsiya avtomata v dvukhkomponentnuyu set' s ogranicheniem na vnutrennie svyazi [Decomposition of an automaton into a two-component net with restrictions on internal communication]. Avtomatika i Vychislitel'naya Tekhnika, 1982, no. 6, pp. 25–32. (in Russian)

16. *Zakrevskij A., Pottosin Yu., and Cheremisinova L.* Optimization in Boolean Space. Tallinn, TUT Press, 2009.
17. *Zakrevskij A., Pottosin Yu., and Cheremisinova L.* Combinatorial Algorithms of Discrete Mathematics. Tallinn, TUT Press, 2008.
18. *Pottosin Yu. V., Toropov N. R., and Shestakov E. A.* Metod minimizatsii sistemy ne polnost'yu opredelennykh bulevykh funktsiy [A method for minimization of a system of incompletely specified Boolean functions]. Informatika, 2009, no. 3(23), pp. 16–26. (in Russian)
19. *Pottosin Yu.* Race-free state assignment for low power asynchronous automaton. Further Improvements in the Boolean Domain. Ed. B. Steinbach. Cambridge Scholars Publ., 2018, pp. 253–267.
20. *Zakrevskiy A. D.* Optimizatsiya pokrytiy mnozhestv [Optimization of set covering]. Logicheskiy Yazyk dlya Predstavleniya Algoritmov Sinteza Releynykh Ustroystv [Logical Language for Representation of Algorithms for Relay Devices Synthesis]. Ed. M. A. Gavrilov. Moscow, Nauka, 1966, pp. 136–148. (in Russian)

ПРИКЛАДНАЯ ТЕОРИЯ ГРАФОВ

УДК 519.1, 519.8 DOI 10.17223/20710410/70/3

О ЦЕЛОЧИСЛЕННОМ ЛИНЕЙНОМ ПРОГРАММИРОВАНИИ ДЛЯ ЗАДАЧ КЛАСТЕРИЗАЦИИ ВЕРШИН ГРАФА[1]

А. В. Моршинин

Институт математики им. С. Л. Соболева СО РАН, г. Омск, Россия

E-mail: morshinin.alexander@gmail.com

Задачи кластеризации являются важной частью анализа данных. В них требуется разбить заданное множество объектов на несколько подмножеств (кластеров) на основе сходства объектов друг с другом. Задача кластеризации вершин графа является формализацией задачи кластеризации. Сходство объектов задаётся с помощью рёбер некоторого графа, вершины которого взаимно однозначно соответствуют объектам. Существует множество вариантов задачи: с ограничением на число и размер кластеров, взвешенные и ориентированные постановки; все известные варианты задачи являются NP-трудными. Данная работа посвящена одному из подходов к решению задачи — построению моделей целочисленного линейного программирования. Приведён обзор известных и предложены новые подходы к построению таких моделей. Новые модели могут использоваться как для нахождения точных решений, так и для построения приближённых алгоритмов. Проведён вычислительный эксперимент, направленный на оценку времени, необходимого алгоритмам, опирающимся на различные модели, для нахождения точного решения. Показано, что один из алгоритмов, опирающихся на новые модели, быстрее других находит решение для задачи с ограниченным числом кластеров.

Ключевые слова: *кластерный граф, целочисленное линейное программирование, NP-трудная задача.*

ON AN INTEGER LINEAR PROGRAMMING FOR CORRELATION CLUSTERING

A. V. Morshinin

Sobolev Institute of Mathematics SB RAS, Omsk, Russia

Clustering problems form an important section of data analysis. In these problems, we need to partition a given set of objects into several subsets (clusters) based on the similarity of the objects to each other. Correlation clustering is a formalization of the clustering problem. Similar objects are connected by edges of a graph and vertices are in one-to-one correspondence with the objects. The problem has several variants: with limited number and size of clusters, weighted, and directed. All known variants are NP-hard. We investigate an approach to solve the problems that involves

[1]Исследование выполнено за счёт гранта РНФ № 22-71-10015.

building integer linear programming models. We review existing models and propose new approaches to model building. The new models can be used both to find exact solutions and to construct approximate algorithms. An experimental study has been conducted to evaluate the computation time to find exact solutions by algorithms based on different models. It showed that one of the algorithms based on the new models is faster than others in finding solutions for a variant of the problem with a limited number of clusters.

Keywords: *cluster graph, integer linear programming, NP-hard problem.*

Введение

В *задаче кластеризации* требуется разбить заданное множество объектов на подмножества (кластеры) на основе сходства между объектами. Одной из наиболее наглядных формализаций этой задачи является *задача кластеризации вершин графа* [1] (*correlation clustering* [2], *cluster editing* [3, 4] и др.), где сходство объектов задаётся рёбрами графа, вершины которого взаимно однозначно соответствуют объектам.

Будем рассматривать только неориентированные графы без петель и кратных рёбер, то есть *обыкновенные графы*. Обыкновенный граф $G = (V, E)$ называется *кластерным*, если каждая его компонента связности является полным графом.

Если $G_1 = (V, E_1)$ и $G_2 = (V, E_2)$ — помеченные графы на одном и том же множестве вершин V, то *расстояние* $\mathrm{d}(G_1, G_2)$ между ними определяется как

$$\mathrm{d}(G_1, G_2) = |E_1 \setminus E_2| + |E_2 \setminus E_1|,$$

то есть $\mathrm{d}(G_1, G_2)$ равно числу различающихся рёбер в графах G_1 и G_2.

Определим следующие множества кластерных графов:

1) $\mathbf{CGS}(V)$ (Cluster Graph Set) — множество всех кластерных графов на множестве вершин V;
2) $\mathbf{CGS}_k(V)$ — множество всех кластерных графов на V, имеющих k компонент связности $(1 \leqslant k \leqslant |V|)$;
3) $\mathbf{CGS}_{\leqslant k}(V)$ — множество всех кластерных графов на V, имеющих не более k компонент связности $(1 \leqslant k \leqslant |V|)$. Очевидно, что $\mathbf{CGS}_{\leqslant k}(V) = \bigcup\limits_{i=1}^{k} \mathbf{CGS}_i(V)$.

Эти множества тесно связаны со следующими минимизационными вариантами задачи кластеризации вершин графа:

— MIN-DISAGREE. Для произвольного графа $G = (V, E)$ найти ближайший к G кластерный граф $C^* \in \mathbf{CGS}(V)$, то есть граф, для которого величина $\mathrm{d}(G, C^*)$ минимальна среди всех графов из $\mathbf{CGS}(V)$.
— MIN-DISAGREE$_k$. Для произвольного графа $G = (V, E)$ и целого числа k, $2 \leqslant k \leqslant |V|$, найти ближайший к G кластерный граф $C^* \in \mathbf{CGS}_k(V)$.
— MIN-DISAGREE$_{\leqslant k}$ формулируется аналогично.

Изучение задач кластеризации вершин графа имеет множество приложений. P. Solé и T. Zaslavsky [5] показали связь этих задач с теорией кодирования; R. Shamir, R. Sharan и D. Tsur [3], а также A. Ben-Dor, R. Shamir и Z. Yakhimi [4] — с вычислительной биологией. Изучая задачи классификации документов, N. Bansal, A. Blum и S. Chawla [2] фактически переоткрыли эти задачи. Кластеризация вершин графа связана с такими проблемами, как кластеризация многомерных данных [6], бикластеризация [7] и др.

Вычислительная сложность задач кластеризации вершин графа долгое время оставалась неизвестной. В 1986 г. M. Křivánek и J. Morávek [8] доказали, что задача MIN-DISAGREE является NP-трудной, однако их работа осталась незамеченной. В 2004 г. N. Bansal, A. Blum и S. Chawla [2] и независимо R. Shamir, R. Sharan и D. Tsur [3] показали NP-трудность задачи MIN-DISAGREE. В [2] также доказано, что задача MIN-DISAGREE$_k$ является NP-трудной при любом фиксированном $k \geqslant 2$; в 2006 г. I. Giotis и V. Guruswami [9] опубликовали более простое доказательство этого результата. В том же году А. А. Агеев, В. П. Ильев, А. В. Кононов и А. С. Талевнин [10] доказали, что задачи MIN-DISAGREE$_2$ и MIN-DISAGREE$_{\leqslant 2}$ NP-трудны уже на кубических графах, откуда вывели, что все упомянутые ранее задачи кластеризации вершин графа являются NP-трудными, включая задачу MIN-DISAGREE$_{\leqslant k}$.

Известно множество приближённых алгоритмов для задач кластеризации вершин графа. В [2] представлен 3-приближённый алгоритм для задачи MIN-DISAGREE$_{\leqslant k}$; в [10] доказано существование рандомизированной полиномиальной приближённой схемы для задачи MIN-DISAGREE$_{\leqslant 2}$, а в [9] предложена рандомизированная полиномиальная приближённая схема для задачи MIN-DISAGREE$_{\leqslant k}$ (для любого фиксированного $k \geqslant 2$). Указав, что сложность схемы из [9] лишает её перспективы практического применения, T. Coleman, J. Saunderson и A. Wirth [11] в 2008 г. разработали 2-приближённый алгоритм решения задачи MIN-DISAGREE$_{\leqslant 2}$, применив процедуру локального поиска к каждому допустимому решению, полученному с помощью 3-приближённого алгоритма из [2]. Для задачи MIN-DISAGREE$_2$ В. П. Ильев, С. Д. Ильева и А. А. Навроцкая [12] в 2011 г. предложили 3-приближённый алгоритм, а в 2020 г. В. П. Ильев, С. Д. Ильева и А. В. Моршинин [13] усилили этот результат, предложив 2-приближённый алгоритм. Эти же авторы в [14] представили два 6-приближённых алгоритма для задачи MIN-DISAGREE$_{\leqslant 3}$.

Что касается задачи MIN-DISAGREE, то в 2005 г. M. Charikar, V. Guruswami и A. Wirth [15] показали, что она является APX-трудной, и разработали 4-приближённый алгоритм её решения, опирающийся на *модель целочисленного линейного программирования* (ЦЛП). В 2008 г. N. Ailon, M. Charikar и A. Newman [16] предложили 2,5-приближённый алгоритм для задачи MIN-DISAGREE. В 2015 г. S. Chawla, K. Makarychev, T. Schramm и G. Yaroslavtsev [17] разработали для этой задачи 2,06-приближённый алгоритм.

Настоящая работа посвящена построению моделей ЦЛП для задач кластеризации вершин графа. В п. 1 приводится обзор известного подхода к построению моделей ЦЛП для рассматриваемых задач и предлагается простой способ сокращения количества переменных и ограничений модели. В п. 2 описаны новые подходы к построению моделей ЦЛП, показано, что один из них позволяет значительно сократить количество ограничений для варианта задачи с ограниченным числом кластеров. В п. 3 приведены результаты экспериментального исследования времени работы точных алгоритмов, опирающихся на известные и новые модели ЦЛП.

1. Известные модели ЦЛП для задач кластеризации вершин графа

1.1. Характеризация множеств **CGS**, **CGS**$_k$ и **CGS**$_{\leqslant k}$ запрещёнными графами

Модели ЦЛП, представленные здесь, опираются на характеризацию кластерных графов запрещёнными графами. Множество кластерных графов может быть описано конечным множеством *запрещённых графов*, которые не могут содержаться в качестве (порождённых) подграфов ни в одном из графов данного множества.

Известно, что граф $G = (V, E)$ принадлежит множеству $\mathbf{CGS}(V)$ тогда и только тогда, когда он не содержит в качестве порождённого подграфа простую цепь P_2 [18]. Легко проверить, что граф $G = (V, E)$ принадлежит множеству $\mathbf{CGS}_{\leqslant k}(V)$ тогда и только тогда, когда он принадлежит множеству $\mathbf{CGS}(V)$ и не содержит в качестве порождённого подграфа пустой граф O_{k+1}.

Сложнее определить принадлежность графа к множеству $\mathbf{CGS}_k(V)$. Помимо ограничения количества кластеров сверху, необходимо также ограничить их количество снизу. Используем следующее определение: *звёздный лес* SF графа $G = (V, E)$ — это остовный подграф графа G, каждая компонента связности которого является звездой [19] (рис. 1); SF_k обозначает звёздный лес, содержащий ровно k звёзд.

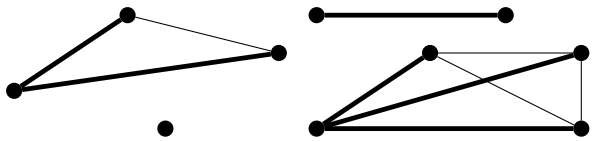

Рис. 1. Звёздный лес графа G выделен жирным. Очевидно, что $G \notin \mathbf{CGS}_i(V), i = 1, 2, 3$

Утверждение 1. Граф $G = (V, E)$ принадлежит множеству $\mathbf{CGS}_k(V)$ тогда и только тогда, когда:

1) $G \in \mathbf{CGS}_{\leqslant k}(V)$;
2) G не содержит в качестве подграфа звёздный лес SF_{k-1}.

Доказательство.

Необходимость. Пусть $G \in \mathbf{CGS}_k(V)$. Очевидно, что $G \in \mathbf{CGS}_{\leqslant k}(V)$. Поскольку G имеет ровно k компонент связности и каждая компонента содержит как минимум одну звезду в качестве остовного подграфа, то минимальный звёздный лес в G состоит ровно из k звёзд.

Достаточность. Пусть $G \in \mathbf{CGS}_{\leqslant k}(V)$ и G не содержит в качестве подграфа звёздный лес SF_{k-1}. По определению множества $\mathbf{CGS}_{\leqslant k}(V)$, граф G принадлежит $\mathbf{CGS}_j(V)$ для некоторого $1 \leqslant j \leqslant k$. Предположим, что $j < k$. Тогда, как доказано ранее, G содержит некоторый звёздный лес SF_j. Если $j = k - 1$, то G содержит звёздный лес SF_{k-1}. Пусть $j < k - 1$. Поскольку пустой граф O_1 является звездой, сделаем следующую процедуру: в SF_j возьмём любую вершину степени 1 и удалим ребро, одним из концов которого является эта вершина. Полученный таким образом граф будет звёздным лесом G с $j + 1$ звездой. Последовательно повторяя процедуру $k - 1 - j > 0$ раз, можно построить звёздный лес SF_{k-1}. Полученное противоречие доказывает, что $k = j$, а значит, $G \in \mathbf{CGS}_k(V)$.

Утверждение 1 доказано. ■

1.2. Модели ЦЛП для задачи MIN-DISAGREE

Рассмотрим произвольный граф $G = (V, E)$ с n вершинами. В [15] предложена модель ЦЛП для задачи MIN-DISAGREE. Кластеризация вершин может быть представлена с помощью бинарных переменных x_{ij}, определённых для всех пар вершин i и j, где

$$x_{ij} = \begin{cases} 0, & \text{если вершины } i \text{ и } j \text{ принадлежат одному кластеру,} \\ 1 & \text{иначе.} \end{cases}$$

По умолчанию будем считать, что $x_{ii} = 0$. Из транзитивности отношения принадлежности к кластеру следует, что если $x_{ij} = 0$ и $x_{jr} = 0$, то $x_{ir} = 0$. Это свойство обеспечивается неравенствами треугольника:

$$x_{ir} \leqslant x_{ij} + x_{jr} \quad \text{для всех } i, j, r \in V.$$

Эти неравенства гарантируют, что результирующий кластерный граф не содержит простую цепь P_2 в качестве порожденного подграфа. D. F. Wahid и E. Hassini в литературном обзоре задач кластеризации [20] в явном виде записали ограничение, отражающее неориентированность исходного графа:

$$x_{ij} = x_{ji} \quad \text{для всех } i, j \in V.$$

Таким образом, получаем следующую модель ЦЛП для задачи MIN-DISAGREE:

$$\sum_{ij \in E} x_{ij} + \sum_{ij \notin E} (1 - x_{ij}) \to \min; \tag{1}$$

$$x_{ir} \leqslant x_{ij} + x_{jr} \quad \text{для всех } i, j, r \in V; \tag{2}$$

$$x_{ij} = x_{ji} \quad \text{для всех } i, j \in V; \tag{3}$$

$$x_{ij} \in \{0, 1\} \quad \text{для всех } i, j \in V. \tag{4}$$

Легко видеть, что для каждой пары вершин необходимо две симметричные переменные, для каждой тройки вершин — шесть неравенств треугольника. Чтобы избавиться от симметричных переменных, запишем вместо неравенства треугольника для каждой упорядоченной тройки вершин три неравенства треугольника для каждой неупорядоченной тройки вершин:

$$\sum_{ij \in E} x_{ij} + \sum_{ij \notin E} (1 - x_{ij}) \to \min; \tag{5}$$

$$\begin{cases} x_{ir} \leqslant x_{ij} + x_{jr} \\ x_{ij} \leqslant x_{ir} + x_{jr} \\ x_{jr} \leqslant x_{ij} + x_{ir} \end{cases} \quad \text{для всех } i, j, r \in V; \tag{6}$$

$$x_{ij} \in \{0, 1\} \quad \text{для всех } i, j \in V. \tag{7}$$

Обе модели содержат $O(n^2)$ переменных и $O(n^3)$ ограничений, но модель (5)–(7) содержит в 2 раза меньше переменных и ограничений, чем модель (1)–(4).

1.3. Модели ЦЛП для задачи MIN-DISAGREE$_{\leqslant k}$ и MIN-DISAGREE$_k$

Для построения модели ЦЛП для задачи MIN-DISAGREE$_{\leqslant k}$ достаточно дополнить базовые модели (1)–(4) и (5)–(7) следующим ограничением:

$$x_{i_1 i_2} + \ldots + x_{i_k i_{k+1}} \leqslant (k+2)(k-1)/2 \quad \text{для всех } i_1, \ldots, i_{k+1} \in V. \tag{8}$$

Это неравенство гарантирует, что в любом подмножестве из $(k+1)$ вершин хотя бы одна пара принадлежит одному кластеру, что исключает появление пустого подграфа O_{k+1}.

Обе расширенные модели (1)–(4),(8) и (5)–(8) содержат $O(n^{k+1})$ ограничений типа (8). Однако модель (1)–(4),(8) требует примерно в $(k+1)!$ раз больше таких ограничений, чем модель (5)–(8), из-за необходимости учитывать все перестановки вершин.

В задаче MIN-DISAGREE$_k$ необходимо ограничить количество кластеров снизу. Утверждение 1 позволяет легко ввести это ограничение для $k = 2$, а именно: необходимо, чтобы G не содержал в качестве подграфа звёздный лес SF_1. Для этого достаточно, чтобы не все вершины графа G были смежны с вершиной 1. Это утверждение можно записать в виде следующего неравенства:

$$\sum_{j=2}^{n} x_{1j} \geqslant 1. \tag{9}$$

Однако для случая $k \geqslant 3$ построение аналогичных компактных линейных ограничений представляет значительную сложность. В п. 2 представлен новый подход к построению моделей ЦЛП для этих задач, который позволяет простым способом ограничить количество кластеров как сверху, так и снизу.

2. Новые модели ЦЛП для задач кластеризации вершин графа

2.1. Анализ неравенства треугольника

Как было отмечено ранее, для каждой тройки вершин модели (1)–(4) и (5)–(7) содержат соответственно 6 и 3 неравенства треугольника. Рассмотрим следующий подход к сокращению их количества.

Ключевое наблюдение: в моделях (1)–(4) и (5)–(7) неравенство треугольника запрещает появление P_2 как порождённого подграфа. Для любой тройки вершин сумме $S = x_{ij} + x_{ir} + x_{jr}$ соответствует (рис. 2):

1) $S = 3$: пустой граф O_3;
2) $S = 2$: граф $K_2 \cup O_1$;
3) $S = 1$: простая цепь P_2 (запрещенная конфигурация);
4) $S = 0$: полный граф K_3 (все вершины в одном кластере).

Рис. 2. Все неизоморфные графы с тремя вершинами

Для исключения запрещённого случая $S = 1$ применим метод линеаризации из работы [21]. Введём условие

$$|x_{ij} + x_{ir} + x_{jr} - 1| \geqslant \varepsilon$$

для некоторого небольшого $\varepsilon > 0$ (например, $\varepsilon = 10^{-3}$).

Используя бинарные переменные $y_{ijr} \in \{0, 1\}$, связывающие две области допустимых значений, и большое число M (например, $M = 10^3$), получаем

$$\sum_{ij \in E} x_{ij} + \sum_{ij \notin E} (1 - x_{ij}) \to \min; \tag{10}$$

$$x_{ij} + x_{ir} + x_{jr} - 1 \geqslant \varepsilon - (1 - y_{ijr})M \quad \text{для всех } i, j, r \in V; \tag{11}$$

$$x_{ij} + x_{ir} + x_{jr} - 1 \leqslant -\varepsilon + y_{ijr}M \quad \text{для всех } i, j, r \in V; \tag{12}$$

$$x_{ij} \in \{0, 1\} \quad \text{для всех } i, j \in V\ , \tag{13}$$

$$y_{ijr} \in \{0, 1\} \quad \text{для всех } i, j, r \in V. \tag{14}$$

Переменная y_{ijr} равна 1, если $x_{ij} + x_{ir} + x_{jr} - 1 \geqslant \varepsilon$, иначе она равна 0.

Для каждой тройки вершин вместо шести неравенств в модели (1)–(4) и трёх неравенств в модели (5)–(7) имеется два неравенства. Нам пришлось добавить $O(n^3)$ переменных. Эта модель имеет иную структуру области допустимых решений, что является хорошим аргументом для её дальнейшего теоретического исследования (например, для построения приближённых алгоритмов). Однако данный подход не решает фундаментальную проблему поиска точных решений: количество неравенств (8) всё ещё быстро растёт. Это требует разработки новых методов построения моделей ЦЛП.

2.2. Модели ЦЛП для задач MIN-DISAGREE$_{\leqslant 2}$ и MIN-DISAGREE$_2$

Идея работы с абсолютными величинами, применённая для модели (10)–(14), может быть эффективно использована для построения моделей ЦЛП другого типа. Рассмотрим задачу MIN-DISAGREE$_{\leqslant 2}$.

Для каждой вершины $i \in V$ введём бинарную переменную x_i:

$$x_i = \begin{cases} 0, & \text{если вершина } i \text{ принадлежит первому кластеру,} \\ 1, & \text{если вершина } i \text{ принадлежит второму кластеру.} \end{cases}$$

Заметим, что если для вершин $i, j \in V$ в графе G существует ребро $ij \in E$, то выражение $|x_i - x_j|$ равно 0 при $x_i = x_j$, иначе оно равно 1. Если для этих вершин не существует ребра $ij \notin E$, то выражение $|x_i + x_j - 1|$ равно 0 при $x_i \neq x_j$, иначе оно равно 1. Используем данный факт для построения модели целочисленного программирования (ЦП):

$$\sum_{ij \in E} |x_i - x_j| + \sum_{ij \notin E} |x_i + x_j - 1| \to \min,$$
$$x_i \in \{0, 1\} \text{ для всех } i \in V.$$

Эта модель не является линейной. Применим известный приём, который сделает её линейной [22]. Представим каждый модуль в целевой функции в виде суммы двух бинарных переменных $u_{ij} + v_{ij}$, а выражение под модулем — в виде разности этих переменных $v_{ij} - u_{ij}$ $(u_{ij} \cdot v_{ij} = 0)$:

$$\sum_{i,j \in V} u_{ij} + v_{ij} \to \min; \tag{15}$$
$$x_i - x_j + u_{ij} - v_{ij} = 0 \text{ для всех } i, j \in V,\ ij \in E; \tag{16}$$
$$x_i + x_j - 1 + u_{ij} - v_{ij} = 0 \text{ для всех } i, j \in V,\ ij \notin E; \tag{17}$$
$$x_i \in \{0, 1\} \text{ для всех } i \in V; \tag{18}$$
$$u_{ij} \in \{0, 1\} \text{ для всех } i, j \in V; \tag{19}$$
$$v_{ij} \in \{0, 1\} \text{ для всех } i, j \in V. \tag{20}$$

Эта модель содержит $O(n^2)$ переменных и $O(n^2)$ ограничений. Легко получить модель ЦЛП для задачи MIN-DISAGREE$_2$, добавив всего два ограничения. Они гарантируют, что не все вершины одновременно принадлежат одному кластеру:

$$\sum_{i \in V} x_i \geqslant 1; \tag{21}$$
$$\sum_{i \in V} x_i \leqslant n - 1. \tag{22}$$

2.3. Модели ЦЛП для задач MIN-DISAGREE$_{\leqslant k}$ и MIN-DISAGREE$_k$

Для задач с ограничением на число кластеров при $k \geqslant 3$ нельзя использовать одну бинарную переменную для описания принадлежности вершин к кластерам. Вместо этого используем *унитарный код*. Для каждой вершины $i \in V$ и кластера $r \in \{1, \ldots, k\}$ введем бинарную переменную x_{ir}:

$$x_{ir} = \begin{cases} 1, & \text{если вершина } i \text{ принадлежит кластеру } r, \\ 0 & \text{иначе.} \end{cases}$$

Таким образом, для каждой вершины $i \in V$ существует унитарный код $(x_{i1}, \ldots, x_{ik})$, в котором лишь одна координата равна 1.

Заметим, что при таком кодировании если для вершин $i, j \in V$ существует ребро $ij \in E$, то сумма $\frac{1}{2}\sum\limits_{r=1}^{k} |x_{ir} - x_{jr}|$ равна 0 тогда и только тогда, когда вершины i и j принадлежат одному кластеру, иначе она равна 1. С другой стороны, если $ij \notin E$, то значение $\frac{1}{2}\left(\sum\limits_{r=1}^{k} |x_{ir} + x_{jr} - 1| - k + 2\right)$ равно 0 тогда и только тогда, когда вершины i и j принадлежат разным кластерам, иначе оно равно 1.

Действительно, если вершины i и j принадлежат одному кластеру, то им соответствуют одинаковые унитарные коды, если разным, то их унитарные коды отличаются в двух координатах. В первом случае, покомпонентно вычитая векторы и беря полученную разность по модулю, мы либо получим вектор из 0 (i и j принадлежат одному кластеру), либо вектор только с двумя 1 (i и j принадлежат разным кластерам). Аналогично во втором случае: мы либо получим вектор из 1 (i и j принадлежат одному кластеру), либо вектор только с двумя 0 (i и j принадлежат разным кластерам). Дальнейшие вычисления тривиальны.

Используя замену, аналогичную замене для модели (15)–(20), мы получаем следующую модель ЦЛП для задачи MIN-DISAGREE$_{\leqslant k}$:

$$\sum_{i,j \in V} \sum_{r=1}^{k} u_{ijr} + v_{ijr} \to \min; \tag{23}$$

$$x_{ir} - x_{jr} + u_{ijr} - v_{ijr} = 0 \quad \text{для всех } i, j \in V,\ ij \in E,\ r \in \{1, \ldots, k\}; \tag{24}$$

$$x_{ir} + x_{jr} - 1 + u_{ijr} - v_{ijr} = 0 \quad \text{для всех } i, j \in V,\ ij \notin E,\ r \in \{1, \ldots, k\}; \tag{25}$$

$$\sum_{r=1}^{k} x_{ir} = 1 \quad \text{для всех } i \in V,\ r \in \{1, \ldots, k\}; \tag{26}$$

$$x_{ir} \in \{0, 1\} \quad \text{для всех } i \in V,\ r \in \{1, \ldots, k\}; \tag{27}$$

$$u_{ijr} \in \{0, 1\} \quad \text{для всех } i, j \in V,\ r \in \{1, \ldots, k\}; \tag{28}$$

$$v_{ijr} \in \{0, 1\} \quad \text{для всех } i, j \in V,\ r \in \{1, \ldots, k\}. \tag{29}$$

Равенство (26) означает, что одна вершина может принадлежать одному кластеру.

В отличие от моделей (1)–(4),(8), (5)–(8) и (10)–(14),(8), легко построить модель ЦЛП для задачи MIN-DISAGREE$_k$. Для этого достаточно добавить неравенства, гарантирующие, что в каждом кластере есть хотя бы одна вершина:

$$\sum_{i \in V} x_{ir} \geqslant 1 \quad \text{для всех } r \in \{1, \ldots, k\}. \tag{30}$$

Заметим, что модели (23)–(29) и (23)–(30) содержат $O(kn^2)$ переменных и $O(kn^2)$ ограничений. Таким образом, полученные модели имеют значительно меньше ограничений, хотя количество переменных в $O(k)$ раз больше.

3. Экспериментальное исследование

3.1. Описание вычислительного эксперимента

Все перечисленные модели ЦЛП могут быть интересны для построения приближённых алгоритмов, которые могут стать объектом дальнейшего исследования. Мы же сосредоточимся на исследовании времени работы точных алгоритмов, опирающихся на описанные модели, при нахождении оптимальных решений на графах малой размерности. Такое исследование интересно по двум причинам. Во-первых, существуют практические задачи, в которых количество объектов не может быть слишком велико. Таковой является, например, задача разделения небольшой социальной группы (рабочий коллектив, школьный класс и др.) на подгруппы, максимизирующая взаимную симпатию внутри подгрупп [23]. Очевидно, что предпочтительно уметь находить оптимальное решение для групп как можно большего размера. Во-вторых, нахождение точных решений позволяет проводить предварительный анализ приближённых алгоритмов, направленный на построение статистических оценок точности. Здесь также чем большую задачу можно решить оптимально, тем качественнее будет предварительный анализ. При этом для нахождения «хороших» допустимых решений на графах большей размерности можно использовать приближённые алгоритмы, описанные во введении.

Для сравнения времени работы точных алгоритмов, опирающихся на различные модели ЦЛП, проведён вычислительный эксперимент. Цель эксперимента заключалась в том, чтобы на основе статистических данных сравнить различные точные алгоритмы между собой и выявить лучший из них для каждой из задач. Все описанные модели реализованы с помощью языка программирования Python и его библиотеки Python-MIP. В качестве решателя выбран IBM ILOG CPLEX. Вычисления производились на NEOS Server с четырьмя ядрами центрального процессора. Выбор библиотеки Python-MIP мотивирован простотой записи реализованной модели в файл MPS-формата, который отправлялся на NEOS Server. Весь необходимый код доступен по ссылке `github.com/BIGADIL/graph_correlation_clustering_mip`.

Опишем схему проведения вычислительного эксперимента:

1) вводится вероятностное распределение на множестве входов исследуемой задачи, то есть задаётся вероятностное пространство на множестве графов;
2) в соответствии с вероятностным распределением проводится случайный выбор N графов, на которых исследуемыми алгоритмами решается задача;
3) для каждого полученного решения вычисляется время работы, которое является случайной величиной;
4) на основе статистических данных вычисляются оценка математического ожидания времени работы и его доверительный интервал для исследуемых алгоритмов.

Итак, введём на множестве всех графов вероятностное распределение. Для этого зафиксируем параметр $p \in (0, 1)$. Случайный n-вершинный граф $G = (V, E)$ будем получать с использованием следующей процедуры. Для каждой пары вершин (u, v) проводится независимый случайный эксперимент, исходами которого будут наличие ребра uv с вероятностью p и отсутствие ребра с вероятностью $1 - p$. Таким образом,

граф G можно рассматривать как случайный вектор, каждая координата которого соответствует паре вершин графа G. Семейство n-вершинных графов с введённым таким образом распределением обозначается $G(n,p)$ и используется как при теоретическом изучении графов, так и в экспериментальных исследованиях (модель Эрдеша — Реньи) [24].

Параметр p в вероятностной модели представляет собой математическое ожидание плотности случайного графа $G=(V,E)$, которая определяется как $2|E|/n(n-1)$. В экспериментах использовались значения p из множества $\{0{,}33,\ 0{,}5,\ 0{,}67\}$.

Рассмотрим теперь параметр n семейства $G(n,p)$. Главным ограничением на количество вершин в тестовых задачах была сложность отыскания оптимального значения целевой функции. В рамках эксперимента для каждой пары значений n и p было решено по 100 задач. Если для какого-то значения n из 100 тестовых задач исследуемый алгоритм не справлялся хотя бы с 3 % задач за 6500 с, такое значение n не участвовало в исследовании. Также соблюдалось ограничение NEOS Server на размер модели, равное 16,5 Мбайт. Таким образом, для каждого алгоритма и для каждого значения p было подобрано своё граничное значение n.

В качестве оценки математического ожидания времени работы алгоритма при заданных значениях параметров n и p взято его среднее время работы по серии задач при этих параметрах. Поскольку распределение времени работы чаще всего не является нормальным, построение доверительного интервала среднего производилось с помощью процедуры *бутстрепа*. Опишем простейшую реализацию этой процедуры. Из имеющейся выборки генерируется B псевдовыборок того же размера, что и исходная, методом случайного выбора с возвращением. Для каждой псевдовыборки вычисляется псевдостатистика среднего времени работы. После этого псевдостатистики сортируются в порядке возрастания. На уровне значимости α слева и справа отбрасывается по $[\alpha B/2]$ элементов. Среди оставшихся крайние левый и правый элементы являются границами доверительного интервала среднего времени работы. В рамках эксперимента использовалась реализация *метода бутстрепа с коррекцией смещения и ускорением* (BCa) из библиотеки SciPy со значениями параметров $B=10000$ и $\alpha=0{,}05$. Более подробно о процедуре бутстрепа и BCa можно прочитать в [25, 26].

Для исследования выбраны задачи MIN-DISAGREE$_{\leqslant k}$ при $k=2,3$ и MIN-DISAGREE; задача MIN-DISAGREE$_k$ не изучалась. Такой выбор обусловлен следующими факторами:

1) выбранные задачи являются наиболее изученными;
2) для задачи MIN-DISAGREE$_2$ к моделям (1)–(4),(8), (5)–(8) и (10)–(14),(8) необходимо добавить одно ограничение (9), а к модели (15)–(20) — два ограничения (21) и (22), что значительно меньше количества других ограничений. Это дополнение незначительно влияет на общее время решения;
3) для задачи MIN-DISAGREE$_k$, $k\geqslant 3$, к модели (23)–(29) необходимо добавить k ограничений (30), что значительно меньше количества других ограничений. Эти дополнения незначительно влияют на общее время решения. К тому же не существует альтернативы для модели (23)–(30), с которой её можно было бы сравнить.

Модель (1)–(4) эквивалентна модели (5)–(7), но содержит больше переменных и ограничений. Результаты разведочного анализа показали, что алгоритмы, опирающиеся на модель (1)–(4) и её производные, требуют значительно больше времени для достижения оптимума по сравнению с алгоритмами, опирающимися на модель (5)–(7)

и её производные. Так, для задачи MIN-DISAGREE$_{\leqslant 2}$ при значении параметров $n = 35$ и $p = 0{,}33$ точный алгоритм, опирающийся на модель (1)–(4),(8), находит оптимальное решение в среднем за 2807,1 с. При тех же параметрах точный алгоритм, опирающийся на модель (5)–(8), находит точное решение в среднем за 123,1 с, что более чем в 22 раза быстрее. На основании этих результатов было принято решение исключить модель (1)–(4) и её производные из экспериментального исследования.

3.2. Экспериментальное исследование алгоритмов для задачи MIN-DISAGREE$_{\leqslant k}$

Начнём со случая $k = 2$. Обозначим через **TR**, **IN** и **MOD** точные алгоритмы, опирающиеся на модели (5)–(8), (10)–(14),(8) и (15)–(20) соответственно.

Табл. 1 содержит размер (в мегабайтах) каждой из моделей в зависимости от количества вершин. Видно, что модели (5)–(8) и (10)–(14),(8) имеют примерно одинаковый размер и превышают выделенные 16,5 Мбайт при $n = 55$. В свою очередь, модель (15)–(20) остаётся компактной до $n = 95$ включительно. Тем не менее дополнительные ограничения могут сужать область допустимых решений, ускоряя поиск оптимального решения.

Таблица 1
Размер моделей ЦЛП для задачи MIN-DISAGREE$_{\leqslant 2}$

n	(5)–(8)	(15)–(20)	(10)–(14),(8)
20	0,56	0,04	0,60
25	1,13	0,07	1,23
30	2,02	0,10	2,18
35	3,29	0,15	3,55
40	4,98	0,20	5,40
45	7,17	0,25	7,79
50	9,94	0,32	10,81
55	13,36	0,39	14,50
60	17,61	0,46	18,97
65	22,61	0,55	24,37
70	28,46	0,64	30,73
75	35,23	0,74	38,05
80	42,97	0,84	46,47
85	51,75	0,96	55,99
90	61,65	1,07	66,80
95	72,72	1,20	78,83

Среднее время работы точных алгоритмов сильно отличается при $p = 0{,}33\,/\,0{,}5$ и $p = 0{,}67$ (табл. 2–4 и рис. 3).

В случае $p = 0{,}33\,/\,0{,}5$ худшие результаты принадлежат алгоритму **IN**. При $n = 40$ и $p = 0{,}5$ его среднее время работы составляет 3612,7 с, что почти в 4 раза больше, чем у алгоритма **TR**. При $n = 40$ и $p = 0{,}33$ среднее время работы алгоритма **IN** в 1,8 раз больше, чем у алгоритма **TR**, среднее время работы которого равно 812,4 с. Однако оба алгоритма сталкиваются с ограничением по времени при $n = 45$. Лучшие результаты принадлежат алгоритму **MOD**. Лишь при $p = 0{,}33$ и $n = 50$ среднее время его работы приближается к 3600 с и при $n = 55$ сталкивается с ограничением по времени. При $n = 40$ и $p = 0{,}33$ среднее время работы алгоритма **MOD** в 8,4 раз меньше среднего времени работы алгоритма **TR**, а при $n = 40$ и $p = 0{,}5$ — в 5,8 раз. Отметим, что

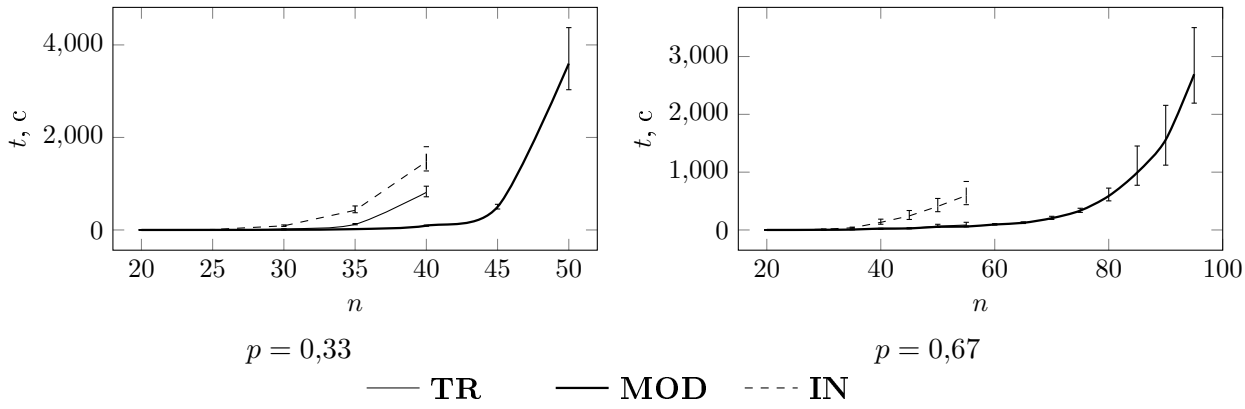

Рис. 3. Среднее время (t, с) работы алгоритмов для задачи MIN-DISAGREE$_{\leqslant 2}$

доверительные интервалы среднего времени работы всех алгоритмов не пересекаются, а значит, среднее время работы алгоритма **MOD** статистически значимо наименьшее.

Таблица 2

Среднее время работы алгоритмов для задачи MIN-DISAGREE$_{\leqslant 2}$, с

n	p								
	0,33			0,5			0,67		
	TR	**MOD**	**IN**	**TR**	**MOD**	**IN**	**TR**	**MOD**	**IN**
20	0,5	0,1	1,8	0,7	0,2	1,7	0,2	0,2	0,6
25	3,7	1,2	9,7	5,2	2,1	12,5	0,9	0,6	2,6
30	21,1	4,5	89,7	43,8	9,7	179,8	3,4	3,1	13,3
35	123,1	18,2	424,8	240,3	36,3	1344,9	7,3	6,7	37,4
40	812,4	96,2	1476,1	952,1	162,9	3612,7	16,6	24,3	131,7
45	–	499,4	–	–	623,7	–	28,8	26,1	243,3
50	–	3598,5	–	–	2153,2	–	63,9	53,1	407,9
55	–	–	–	–	–	–	85,4	58,8	595,2
60	–	–	–	–	–	–	–	94,9	–
65	–	–	–	–	–	–	–	123,5	–
70	–	–	–	–	–	–	–	203,8	–
75	–	–	–	–	–	–	–	334,7	–
80	–	–	–	–	–	–	–	587,8	–
85	–	–	–	–	–	–	–	993,1	–
90	–	–	–	–	–	–	–	1560,5	–
95	–	–	–	–	–	–	–	2694,6	–

В случае $p = 0{,}67$ худшие показатели также принадлежат алгоритму **IN**. При $n = 55$ среднее время его работы составляет 595,2 с. В свою очередь, алгоритмы **TR** и **MOD** при том же значении n тратят в среднем 85,4 и 58,8 с соответственно. Интересно, что доверительные интервалы среднего времени работы алгоритмов **TR** и **MOD** пересекаются до $n = 55$ включительно. Это не позволяет говорить о статистически значимых различиях. Однако при $n = 60$ занимаемый объём памяти становится препятствием к отысканию оптимальных решений для алгоритмов **TR** и **IN**, в то время как алгоритм **MOD** сталкивается с ограничением по времени при $n = 100$.

Таким образом, можно утверждать, что наилучшим алгоритмом для задачи MIN-DISAGREE$_{\leqslant 2}$ является алгоритм **MOD**. Его среднее время работы при $p = 0{,}33 \,/\, 0{,}5$ статистически значимо наименьшее. При $p = 0{,}67$ нет статистически значимых отли-

Таблица 3

Границы доверительного интервала среднего времени работы алгоритмов для задачи MIN-DISAGREE$_{\leqslant 2}$, $p = 0{,}33$ и $p = 0{,}5$, с

n	p					
	0,33			0,5		
	TR	**MOD**	**IN**	**TR**	**MOD**	**IN**
20	[0,4, 0,6]	[0,1, 0,2]	[1,6, 2,1]	[0,6, 0,8]	[0,1, 0,2]	[1,5, 1,9]
25	[3,4, 4,1]	[1,1, 1,2]	[9,2, 10,3]	[4,6, 5,9]	[1,8, 2,3]	[11,3, 13,9]
30	[19,6, 22,9]	[4,2, 4,7]	[76,7, 109,2]	[35,5, 54,7]	[8,5, 10,9]	[145,4, 221,7]
35	[111,8, 138,9]	[16,6, 20,1]	[372,4, 477,2]	[189,1, 303,1]	[30,5, 42,7]	[1030,2, 1776,9]
40	[717,1, 946,1]	[85,7, 109,5]	[1275,5, 1799,9]	[791,1, 1177,3]	[133,6, 200,3]	[2953,3, 4330,3]
45	–	[453,2, 554,5]	–	–	[511,4, 767,8]	–
50	–	[3031,5, 4373,9]	–	–	[1881,5, 2479,4]	–

Таблица 4

Границы доверительного интервала среднего времени работы алгоритмов для задачи MIN-DISAGREE$_{\leqslant 2}$, $p = 0{,}67$, с

n	**TR**	**MOD**	**IN**
20	[0,1, 0,3]	[0,1, 0,2]	[0,5, 0,7]
25	[0,8, 1,2]	[0,5, 0,7]	[2,1, 3,3]
30	[2,7, 4,4]	[2,7, 3,4]	[10,4, 16,8]
35	[5,9, 9,4]	[6,1, 7,6]	[29,7, 49,1]
40	[13,2, 23,8]	[21,1, 28,4]	[98,8, 186,9]
45	[23,1, 37,6]	[22,9, 29,8]	[181,5, 336,5]
50	[46,8, 98,2]	[46,8, 60,2]	[316,1, 545,1]
55	[62,5, 132,7]	[53,9, 67,4]	[437,6, 838,7]
60	–	[84,2, 110,3]	–
65	–	[111,3, 140,3]	–
70	–	[182,3, 236,5]	–
75	–	[304,1, 375,7]	–
80	–	[503,2, 724,1]	–
85	–	[773,2, 1452,2]	–
90	–	[1121,1, 2000,1]	–
95	–	[2193,8, 3500,1]	–

чий в среднем времени работы алгоритмов **TR** и **MOD**, однако последний требует гораздо меньше памяти.

Отметим, что в случае $p = 0{,}33\,/\,0{,}5$ оптимальным решением в 100 % случаев является кластерный граф с двумя кластерами. При $p = 0{,}67$ лишь в 49,5 % случаев оптимальным является граф с двумя кластерами, в оставшихся 51,5 % случаев оптимальный граф содержит один кластер. Это один из факторов, объясняющих лучшее среднее время работы всех алгоритмов на плотных графах.

Перейдём к случаю $k = 3$. В табл. 5 приведены размеры моделей для задачи MIN-DISAGREE$_{\leqslant 3}$ в мегабайтах (Мбайт). Как и в случае $k = 2$, модели (5)–(8) и (10)–(14), (8) имеют примерно одинаковый размер и превышают выделенные 16,5 Мбайт при $n = 40$, модель (23)–(29) занимает немного памяти до $n = 70$ включительно.

Результаты эксперимента представлены в табл. 6–8 и на рис. 4. Худшее среднее время работы в случае $p = 0{,}33\,/\,0{,}5$ принадлежит алгоритму **IN**. При $p = 0{,}5$ и $n = 30$ доверительные интервалы среднего времени работы алгоритмов **IN** и **TR** пересекаются, что не позволяет говорить о статистически значимых отличиях. Однако при

Таблица 5

Размер моделей ЦЛП для задачи MIN-DISAGREE$_{\leqslant 3}$

n	(5)–(8)	(23)–(29)	(10)–(14),(8)
15	0,48	0,07	0,50
20	1,55	0,14	1,60
25	3,92	0,22	4,01
30	8,25	0,33	8,40
35	15,37	0,46	15,64
40	26,48	0,61	26,82
45	42,98	0,79	43,49
50	66,16	0,98	66,87
55	97,47	1,19	98,45
60	138,64	1,43	139,91
65	191,46	1,69	193,18
70	258,65	1,97	260,48

$p = 0{,}33$ их доверительные интервалы среднего времени работы не пересекаются, а при $p = 0{,}33$ и $n = 30$ среднее время работы алгоритма **TR** составляет 1039,1 с, что более чем в 3 раза меньше, чем среднее время работы алгоритма **IN**. Оба алгоритма сталкиваются с ограничением по времени при $n = 35$. Вновь лучшие результаты принадлежат алгоритму **MOD**. Его доверительные интервалы среднего времени работы не пересекаются с доверительными интервалами алгоритмов **IN** и **TR**. Среднее время работы алгоритма **MOD** при $p = 0{,}33$ и $n = 30$ в 5 раз меньше, чем у алгоритма **TR**, а при $p = 0{,}5$ и $n = 30$ — почти в 19 раз.

Таблица 6

Среднее время работы алгоритмов для задачи MIN-DISAGREE$_{\leqslant 3}$, с

n	p								
	0,33			0,5			0,67		
	TR	**MOD**	**IN**	**TR**	**MOD**	**IN**	**TR**	**MOD**	**IN**
15	0,1	0,4	0,7	0,2	0,3	0,7	0,1	0,2	1,3
20	2,6	2,6	18,6	4,7	3,2	20,1	0,4	1,8	4,4
25	51,8	18,1	179,9	104,1	16,5	250,9	2,8	6,2	28,1
30	1039,1	187,1	3282,5	4773,7	254,9	6138,6	21,3	18,9	75,2
35	–	1996,5	–	–	2438,8	–	127,1	61,5	600,8
40	–		–	–	–	–	–	81,6	–
45	–		–	–	–	–	–	242,3	–
50	–	–	–	–	–	–	–	352,1	–
55	–	–	–	–	–	–	–	488,1	–
60	–	–	–	–	–	–	–	877,9	–
65	–	–	–	–	–	–	–	1631,5	–
70	–	–	–	–	–	–	–	2432,2	–

Таблица 7

Границы доверительного интервала среднего времени работы алгоритмов для задачи MIN-DISAGREE$_{\leqslant 3}$, $p = 0{,}33$ и $p = 0{,}5$, с

n	p					
	0,33			0,5		
	TR	**MOD**	**IN**	**TR**	**MOD**	**IN**
15	[0,1, 0,2]	[0,4, 0,5]	[0,6, 1,0]	[0,2, 0,3]	[0,3, 0,4]	[0,6, 1,0]
20	[2,3, 3,0]	[2,5, 2,8]	[16,5, 20,9]	[4,1, 5,5]	[3,1, 3,3]	[18,2, 22,1]
25	[46,1, 59,2]	[16,1, 21,2]	[156,3, 212,6]	[80,3, 142,3]	[15,3, 17,9]	[228,8, 280,0]
30	[914,4, 1216,7]	[171,5, 205,8]	[2824,7, 3977,8]	[3743,3, 6002,5]	[232,2, 280,6]	[5378,5, 7125,8]
35	–	[1813,2, 2254,9]	–	–	[2211,5, 2715,7]	–

Таблица 8

Границы доверительного интервала среднего времени работы алгоритмов для задачи MIN-DISAGREE$_{\leqslant 3}$, $p = 0{,}67$, с

n	**TR**	**MOD**	**IN**
15	[0,1, 0,2]	[0,2, 0,3]	[1,2, 1,5]
20	[0,4, 0,6]	[1,6, 1,9]	[3,4, 6,2]
25	[2,1, 4,1]	[5,7, 6,6]	[20,3, 39,2]
30	[13,8, 41,4]	[17,3, 21,2]	[45,9, 160,8]
35	[86,7, 217,4]	[56,2, 72,1]	[413,0, 983,7]
40	–	[73,8, 92,3]	–
45	–	[185,1, 426,4]	–
50	–	[316,1, 412,5]	–
55	–	[438,4, 567,1]	–
60	–	[787,5, 994,2]	–
65	–	[1405,0, 1942,8]	–
70	–	[2054,9, 2990,1]	–

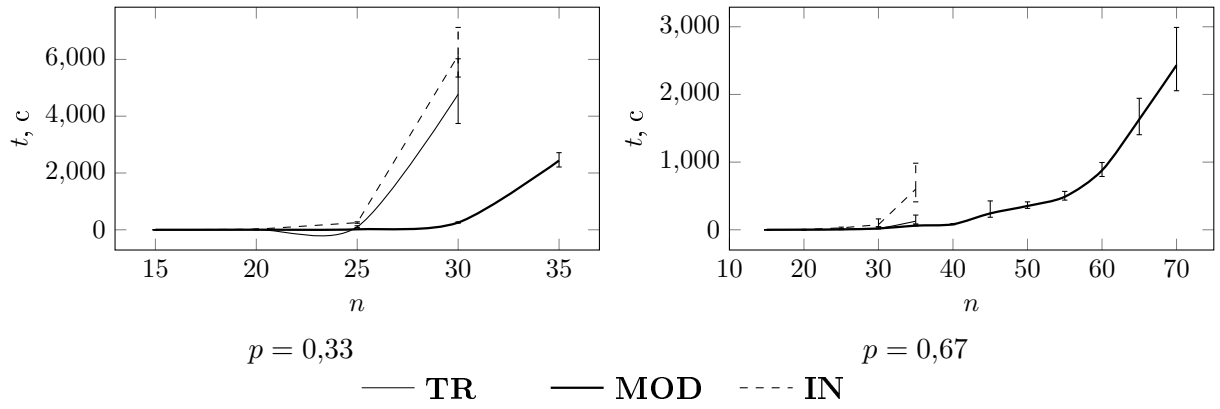

Рис. 4. Среднее время (t, с) работы алгоритмов для задачи MIN-DISAGREE$_{\leqslant 3}$

При $p = 0{,}67$ алгоритм **IN** опять показывает себя хуже алгоритмов **TR** и **MOD**. Интересно, что до $n = 25$ алгоритм **TR** имеет лучшее среднее время работы, чем алгоритм **MOD**, причём их доверительные интервалы не пересекаются. При $n = 30$ среднее время работы алгоритма **MOD** становится меньше среднего времени работы алгоритма **TR**, однако их доверительные интервалы пересекаются. При $n = 35$ доверительные интервалы перестают пересекаться и среднее время работы алгоритма **MOD** становится меньше, чем у алгоритма **TR**. При $n = 40$ опорные модели алгорит-

мов **IN** и **TR** исчерпывают память, а алгоритм **MOD** сталкивается с ограничением по времени при $n = 70$.

Заметим, что результаты эксперимента очень похожи на результаты эксперимента при $k = 2$. При $p = 0{,}33\,/\,0{,}5$ оптимальным решением в 99,4 % случаев является граф с тремя кластерами, а в 0,6 % — с двумя. При $p = 0{,}67$ лишь в 7,4 % случаев оптимальное решение содержит три кластера, в оставшихся случаях — один или два кластера (41 и 51,6 % соответственно).

3.3. Экспериментальное исследование алгоритмов для задачи MIN-DISAGREE

Если в модели (23)–(29) заменить k на n, то получится модель ЦЛП для задачи MIN-DISAGREE. Эта модель содержит $O(n^3)$ переменных и $O(n^3)$ ограничений. Сравним её с другими моделями. Вновь обозначим через **TR**, **IN** и **MOD** точные алгоритмы, опирающиеся на модели (5)–(7), (10)–(14) и (23)–(29) (табл. 9–12).

Таблица 9

Размер моделей ЦЛП для задачи MIN-DISAGREE, Мбайт

n	(5)–(7)	(23)–(29)	(10)–(14)
15	0,15	0,38	0,17
20	0,39	0,96	0,44
25	0,79	1,96	0,90
30	1,42	3,47	1,59
35	2,31	5,64	2,58
40	3,51	8,55	3,93
45	5,05	12,30	5,67
50	7,01	17,07	7,88
55	9,39	22,93	10,58

Таблица 10

Среднее время работы алгоритмов для задачи MIN-DISAGREE, с

n	p								
	0,33			0,5			0,67		
	TR	**MOD**	**IN**	**TR**	**MOD**	**IN**	**TR**	**MOD**	**IN**
15	0,1	1,0	0,1	0,1	1,4	0,4	0,1	0,7	0,1
20	0,7	187,0	1,1	2,3	602,9	5,8	0,2	57,6	0,4
25	4,8	3787,5	9,4	22,6	–	50,2	0,8	2972,9	1,9
30	27,7	–	94,3	841,2	–	1227,3	3,2	–	11,2
35	440,5	–	971,8	–	–	–	6,4	–	19,9
40	–	–	–	–	–	–	19,9	–	264,4
45	–	–	–	–	–	–	58,6	–	866,8
50	–	–	–	–	–	–	204,1	–	–
55	–	–	–	–	–	–	519,3	–	–

Из табл. 9 видно, что для задачи MIN-DISAGREE больше всего памяти требуется модели (23)–(29). При $n = 50$ она занимает более 16,5 Мбайт, в то время как размер моделей (5)–(7) и (10)–(14) растёт медленнее и почти одинаково.

Таблица 11

Границы доверительного интервала среднего времени работы алгоритмов для задачи MIN-DISAGREE, $p = 0{,}33$ и $p = 0{,}5$, с

n	p					
	0,33			0,5		
	TR	**MOD**	**IN**	**TR**	**MOD**	**IN**
15	[0,1, 0,2]	[0,9, 1,2]	[0,1, 0,2]	[0,1, 0,2]	[1,3, 1,6]	[0,3, 0,5]
20	[0,7, 0,9]	[155,2, 231,4]	[0,9, 1,3]	[2,1, 2,5]	[525,3, 698,9]	[5,3, 6,4]
25	[4,3, 5,3]	[2303,9, 5508,2]	[8,3, 10,6]	[20,5, 25,4]	–	[46,2, 54,7]
30	[23,1, 34,8]	–	[79,9, 112,9]	[730,1, 1003,3]	–	[1112,6, 1461,2]
35	[328,1 636,4]	–	[795,4, 1479,3]	–	–	–

При всех значениях параметра p алгоритм **MOD** имеет наихудшее среднее время работы (табл. 10–12 и рис. 5). Так, при $p = 0{,}33$ и $n = 25$ среднее время его работы равно 3785,5 с, что в 789 раз больше, чем у алгоритма **TR**, и в 403 раза больше, чем у алгоритма **IN**. При всех значениях параметра p среднее время работы алгоритма **TR** меньше, чем у алгоритма **IN**. Поскольку доверительные интервалы всех алгоритмов не пересекаются, то среднее время работы алгоритма **TR** статистически значимо наименьшее.

Таблица 12

Границы доверительного интервала среднего времени работы алгоритмов для задачи MIN-DISAGREE, $p = 0{,}67$, с

n	**TR**	**MOD**	**IN**
15	[0,0, 0,1]	[0,6, 0,7]	[0,0, 0,1]
20	[0,2, 0,3]	[45,6, 73,4]	[0,3, 0,6]
25	[0,6, 1,1]	[2457,2, 3697,9]	[1,4, 2,8]
30	[2,3, 4,8]	–	[7,1, 18,1]
35	[4,9, 8,9]	–	[13,8, 36,2]
40	[13,8, 31,7]	–	[125,2, 861,7]
45	[41,3, 87,5]	–	[315,8, 2355,5]
50	[125,7, 469,6]	–	–
55	[292,9, 1288,1]	–	–

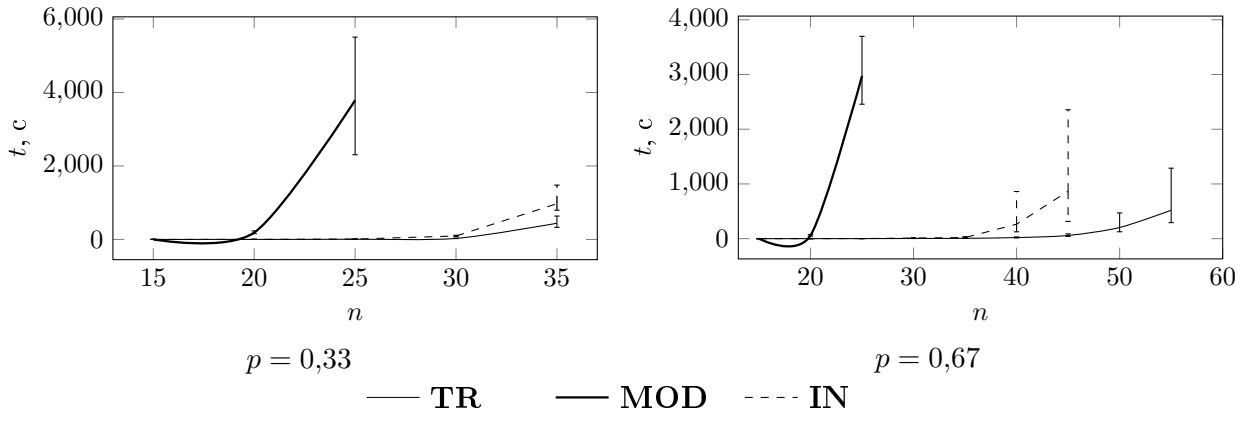

Рис. 5. Среднее время (t, с) работы алгоритмов для задачи MIN-DISAGREE

По результатам исследования можно сделать следующие выводы:

1) в задачах с ограничением числа кластеров наименьшее среднее время достижения оптимума принадлежит алгоритму **MOD**, при этом опорная модель ЦЛП требует небольшой объем памяти. Алгоритмы **TR** и **IN** требуют больше времени и памяти;
2) в задачах без ограничения числа кластеров наименьшее среднее время достижения оптимума принадлежит алгоритму **TR**, при этом опорная модель ЦЛП требует небольшой объем памяти. Алгоритм **IN** требует больше времени, а алгоритм **MOD** — больше времени и памяти;
3) все модели являются перспективными для дальнейшего теоретического исследования из-за разной структуры областей допустимых значений (например, для построения приближённых алгоритмов).

Заключение

В работе рассматриваются задачи кластеризации вершин графа. Изучается подход к построению моделей ЦЛП для этих задач. Приведён обзор известных моделей ЦЛП, а также предложены новые подходы к их построению, один из которых позволяет значительно сократить количество неравенств для задачи с ограничением числа кластеров. Из результатов вычислительного эксперимента следует, что один из алгоритмов, опирающийся на новые модели, является наилучшим при поиске точных решений в задачах с ограничением числа кластеров. В то же время в варианте задачи без ограничений лучшие результаты принадлежат алгоритму, опирающемуся на известную модель. Все описанные модели представляют интерес для теоретического исследования в контексте построения приближённых алгоритмов.

ЛИТЕРАТУРА

1. *Schaeffer S. E.* Graph clustering // Comput. Sci. Rev. 2005. V. 1. No. 1. P. 27–64.
2. *Bansal N., Blum A., and Chawla S.* Correlation clustering // Machine Learning. 2004. V. 56. P. 89–113.
3. *Shamir R., Sharan R., and Tsur D.* Cluster graph modification problems // Discrete Appl. Math. 2004. V. 144. No. 1–2. P. 173–182.
4. *Ben-Dor A., Shamir R., and Yakhimi Z.* Clustering gene expression patterns // J. Comput. Biol. 1999. V. 6. No. 3–4. P. 281–297.
5. *Solé P. and Zaslavsky T.* A coding approach to signed graphs // SIAM J. Discrete Math. 1994. No. 7. P. 544–553.
6. *Kriegel H. P., Kroger P., and Zimek A.* Clustering high-dimensional data // ACM Trans. Knowledge Discovery from Data. 2009. No. 3. P. 1–58.
7. *Balamurugan R., Natarajan A. M., and Premalatha K.* Stellar-mass black hole optimization for biclustering microarray gene expression data // Appl. Artif. Intell. 2015. V. 29. No. 4. P. 353–381.
8. *Křivánek M. and Morávek J.* NP-hard problems in hierarchical-tree clustering // Acta Informatica. 1986. V. 23. P. 311–323.
9. *Giotis I. and Guruswami V.* Correlation clustering with a fixed number of clusters // Theory of Computing. 2006. V. 2. No. 1. P. 249–266.
10. *Агеев А. А., Ильев В. П., Кононов А. В., Талевнин А. С.* Вычислительная сложность задачи аппроксимации графов // Дискретн. анализ и исслед. опер. Сер. 1. 2006. Т. 13. № 1. С. 3–11.

11. *Coleman T., Saunderson J., and Wirth A.* A local-search 2-approximation for 2-correlation-clustering // LNCS. 2008. V. 5193. P. 308–319.
12. *Ильев В. П., Ильева С. Д., Навроцкая А. А.* Приближенные алгоритмы для задач аппроксимации графов // Дискретн. анализ и исслед. опер. 2011. Т. 18. № 1. С. 41–60.
13. *Ильев В. П., Ильева С. Д., Моршинин А. В.* 2-Приближенные алгоритмы для двух задач кластеризации на графах // Дискретн. анализ и исслед. опер. 2020. Т. 27. № 3. С. 88–108.
14. *Ильев В. П., Ильева С. Д., Моршинин А. В.* Алгоритмы приближённого решения одной задачи кластеризации графа // Прикладная дискретная математика. 2019. № 45. С. 64–77.
15. *Charikar M., Guruswami V., and Wirth A.* Clustering with qualitative information // J. Comput. Syst. Sci. 2005. V. 71. No. 3. P. 360–383.
16. *Ailon N., Charikar M., and Newman A.* Aggregating inconsistent information: Ranking and clustering // J. ACM. 2008. V. 55. No. 5. P. 1–27.
17. *Chawla S., Makarychev K., Schramm T., and Yaroslavtsev G.* Near optimal LP algorithm for correlation clustering on complete and complete k-partite graphs // Proc. STOC'15. Portland, Oregon, USA, 2015. P. 219–228.
18. *Nishimura N., Ragde P., and Thilikos D. M.* On graph powers for leaf-labeled trees // J. Algorithms. 2002. V. 42. No. 1. P. 69–108.
19. *Ferneyhough S., Haas R., Hanson D., and MacGillivaray G.* Star forests, dominating sets and Ramsey-type problems // Discrete Math. 2002. V. 245. No. 1–3. P. 255–262.
20. *Wahid D. F. and Hassini E.* A literature review on correlation clustering: cross-disciplinary taxonomy with bibliometric analysis // Oper. Res. Forum. 2020. V. 3. No. 47. P. 1–42.
21. *Wiliams H. P. and Hong Y.* Representations of the all-different predicate of constraint satisfaction in integer programming // NFORMS J. Computing. 2001. V. 13. No. 2. P. 96–103.
22. *Bisschop J.* Aimms Optimization Modeling. `https://documentation.aimms.com/_downloads/AIMMS_modeling.pdf`. 2023.
23. *Harary F.* On the notion of balance of a signed graph // Michigan Math. J. 1953. No. 2. P. 143–146.
24. *Alon N. and Spencer J. H.* The Probabilistic Method. 4th ed. N.Y.: Wiley & Sons, 2016.
25. *Horowitz J. L.* Bootstrap methods in econometrics // Ann. Rev. Economics. 2019. V. 11. P. 193–224.
26. *Efron B.* Better bootstrap confidence intervals // J. ASA. 2014. V. 82. No. 397. P. 171–185.

REFERENCES

1. *Schaeffer S. E.* Graph clustering. Comput. Sci. Rev., 2005, vol. 1, no. 1, pp. 27–64.
2. *Bansal N., Blum A., and Chawla S.* Correlation clustering. Machine Learning, 2004, vol. 56, pp. 89–113.
3. *Shamir R., Sharan R., and Tsur D.* Cluster graph modification problems. Discrete Appl. Math., 2004, vol. 144, no. 1–2, pp. 173–182.
4. *Ben-Dor A., Shamir R., and Yakhimi Z.* Clustering gene expression patterns. J. Comput. Biol., 1999, vol. 6, no. 3–4, pp. 281–297.
5. *Solé P. and Zaslavsky T.* A coding approach to signed graphs. SIAM J. Discrete Math., 1994, no. 7, pp. 544–553.
6. *Kriegel H. P., Kroger P., and Zimek A.* Clustering high-dimensional data. ACM Trans. Knowledge Discovery from Data, 2009, no. 3, pp. 1–58.
7. *Balamurugan R., Natarajan A. M., and Premalatha K.* Stellar-mass black hole optimization for biclustering microarray gene expression data. Appl. Artif. Intell., 2015, vol. 29, no. 4, pp. 353–381.

8. *Křivánek M. and Morávek J.* NP-hard problems in hierarchical-tree clustering. Acta Informatica, 1986, vol. 23, pp. 311–323.
9. *Giotis I. and Guruswami V.* Correlation clustering with a fixed number of clusters. Theory of Computing, 2006, vol. 2, no. 1, pp. 249–266.
10. *Ageev A. A., Il'ev V. P., Kononov A. V., and Talevnin A. S.* Computational complexity of the graph approximation problem. J. Appl. Industr. Math., 2007, vol. 1, no. 1, pp. 1–8.
11. *Coleman T., Saunderson J., and Wirth A.* A local-search 2-approximation for 2-correlation-clustering. LNCS, 2008, vol. 5193, pp. 308–319.
12. *Il'ev V. P., Il'eva S. D., and Navrotskaya A. A.* Approximation algorithms for graph approximation problems. J. Appl. Industr. Math., 2011, vol. 5, no. 4, pp. 569–581.
13. *Il'ev V. P., Il'eva S. D., and Morshinin A. V.* 2-Approximation algorithms for two graph clustering problems. J. Appl. Industr. Math., 2020, vol. 14, no. 3, pp. 490–502.
14. *Il'ev V. P., Il'eva S. D., and Morshinin A. V.* Algoritmy priblizhennogo resheniya odnoy zadachi klasterizatsii grafa [Approximate algorithms for graph clustering problem]. Prikladnaya Diskretnaya Matematika, 2019, no. 45, pp. 64–77. (in Russian)
15. *Charikar M., Guruswami V., and Wirth A.* Clustering with qualitative information. J. Comput. Syst. Sci., 2005, vol. 71, no. 3, pp. 360–383.
16. *Ailon N., Charikar M., and Newman A.* Aggregating inconsistent information: Ranking and clustering. J. ACM, 2008, vol. 55, no. 5, pp. 1–27.
17. *Chawla S., Makarychev K., Schramm T., and Yaroslavtsev G.* Near optimal LP algorithm for correlation clustering on complete and complete k-partite graphs. Proc. STOC'15, Portland, Oregon, USA, 2015, pp. 219–228.
18. *Nishimura N., Ragde P., and Thilikos D. M.* On graph powers for leaf-labeled trees. J. Algorithms, 2002, vol. 42, no. 1, pp. 69–108.
19. *Ferneyhough S., Haas R., Hanson D., and MacGillivaray G.* Star forests, dominating sets and Ramsey-type problems. Discrete Math., 2002, vol. 245, no. 1–3, pp. 255–262.
20. *Wahid D. F. and Hassini E.* A literature review on correlation clustering: cross-disciplinary taxonomy with bibliometric analysis. Oper. Res. Forum, 2020, vol. 3, no. 47, pp. 1–42.
21. *Wiliams H. P. and Hong Y.* Representations of the all-different predicate of constraint satisfaction in integer programming. NFORMS J. Computing, 2001, vol. 13, no. 2, pp. 96–103.
22. *Bisschop J.* Aimms Optimization Modeling. `https://documentation.aimms.com/_downloads/AIMMS_modeling.pdf`, 2023.
23. *Harary F.* On the notion of balance of a signed graph. Michigan Math. J., 1953, no. 2, pp. 143–146.
24. *Alon N. and Spencer J. H.* The Probabilistic Method, 4th ed. N.Y., Wiley & Sons, 2016.
25. *Horowitz J. L.* Bootstrap methods in econometrics. Ann. Rev. Economics, 2019, vol. 11, pp. 193–224.
26. *Efron B.* Better bootstrap confidence intervals. J. ASA, 2014, vol. 82, no. 397, pp. 171–185.

УДК 519.17 DOI 10.17223/20710410/70/4

ОПТИМАЛЬНЫЕ ГРАФЫ С ТОЧКОЙ СОЧЛЕНЕНИЯ И ЗАДАННОЙ РЁБЕРНОЙ СВЯЗНОСТЬЮ

Б. А. Теребин, М. Б. Абросимов

Саратовский национальный исследовательский государственный университет имени Н. Г. Чернышевского, г. Саратов, Россия

E-mail: bogdan.terebin@yandex.ru, mic@rambler.ru

Вершинной связностью k называется наименьшее число вершин, удаление которых приводит к несвязному или тривиальному графу. Рёберной связностью λ нетривиального графа называется наименьшее число рёбер, удаление которых приводит к несвязному графу. Д. Фалкерсон и Л. Шепли решали задачу определения минимального числа рёбер в графе с заданным числом вершин n и с заданной рёберной связностью λ. В работе исследуются минимальные по числу рёбер n-вершинные графы, которые имеют заданные значения вершинной и рёберной связности. Основной результат состоит в том, что определяется минимальное число рёбер, которые могут иметь n-вершинные графы с точкой сочленения и заданной рёберной связностью $\lambda > 1$: $[(\lambda n + \lambda + 1)/2]$. Предлагается схема построения графов с таким числом рёбер. Это всегда возможно при $n \geqslant 2\lambda$.

Ключевые слова: *граф, вершинная связность, рёберная связность.*

OPTIMAL GRAPHS WITH A CUT VERTEX AND GIVEN EDGE CONNECTIVITY

B. A. Terebin, M. B. Abrosimov

Saratov State University, Saratov, Russia

The vertex connectivity k is the smallest number of vertices whose removal leads to a disconnected or trivial graph. The edge connectivity λ of a nontrivial graph is the smallest number of edges whose removal leads to a disconnected graph. D. Fulkerson and L. Shapley solved the problem of determining the minimum number of edges in a graph with a given number of vertices n and a given edge connectivity λ. In this paper, we study the minimal n-vertex graphs with given values of vertex and edge connectivity. The main result is that we determine the minimum number of edges that n-vertex graphs with an articulation point and a given edge connectivity $\lambda > 1$ can have: $[(\lambda n + \lambda + 1)/2]$. A scheme for constructing graphs with such a number of edges is proposed. This is always possible for $n \geqslant 2\lambda$.

Keywords: *graph, vertex connectivity, edge connectivity.*

1. Предварительные результаты

Связным называется граф, любая пара вершин которого соединена путём. В противном случае граф называется *несвязным*. *Тривиальным* называется одновершинный граф. Граф, любые две вершины которого смежны, называется *полным*. В работе рассматриваются простые неориентированные графы. Основные понятия из теории графов используются в соответствии с работами [1, 2].

Определение 1. *Вершинной связностью* k графа G называется наименьшее число вершин, удаление которых приводит к несвязному или тривиальному графу.

Определение 2. *Рёберной связностью* λ нетривиального графа G называется наименьшее число рёбер, удаление которых приводит к несвязному графу.

Обозначим минимальную степень вершины в графе через δ.

Вершинная связность, рёберная связность и минимальная степень вершины δ связаны неравенством Уитни [3].

Теорема 1 [3]. Для любого графа G справедливо неравенство $k \leqslant \lambda \leqslant \delta$.

Г. Чартрэнд и Ф. Харари в работе [4] доказали, что для подходящих значений k, λ и δ существует соответствующий граф:

Теорема 2 [4]. Для любых натуральных чисел a, b, c, таких, что $0 < a \leqslant b \leqslant c$, существует граф G, у которого $k = a$, $\lambda = b$, $c = \delta$.

В работе [5] рассматривается задача о поиске графов с минимальным числом вершин и рёбер для любых a, b, c из теоремы 2. Найдено полное решение этой задачи, причём для всех рассматриваемых наборов значений a, b, c доказываются значения минимального числа вершин и рёбер, а также строятся графы с указанным числом вершин и рёбер. Некоторые из полученных результатов существенным образом используются для решения задачи, которая рассматривается в данной работе: нахождение и описание множеств графов, состоящих из заданного числа вершин n с минимальным числом рёбер для пар возможных значений k и λ. В частности, далее используются следующие результаты из работы [5]:

Теорема 3 [5]. Граф с наименьшим количеством вершин и рёбер, удовлетворяющий условию $a < b = c$, у которого $k = a$, $\lambda = b$, $c = \delta$, является графом с числом вершин $2(c+1) - a$ и числом рёбер $c^2 - a^2 + a + c + \sigma$, где

$$\sigma = \begin{cases} 0, & \text{если } \lceil (2a^2 - ac - 2a)/2 \rceil \leqslant 0, \\ \lceil (2a^2 - ac - 2a)/2 \rceil & \text{иначе.} \end{cases}$$

В [6] Д. Фалкерсон и Л. Шепли рассматривают задачу описания графов с минимальным числом рёбер для заданного числа вершин и рёберной связности $\lambda(G)$. В данной работе мы рассмотрим более общую задачу: для заданного числа вершин n, вершинной связности $k(G)$ и рёберной связности $\lambda(G)$ требуется определить минимальное число рёбер, которое может иметь граф G с указанными параметрами [7, 8]. Согласно неравенству Уитни, минимальная степень вершины в графе не меньше рёберной связности, поэтому очевидно, что в искомом графе не может быть меньше чем $\lceil \lambda n/2 \rceil$ рёбер. Семейство графов именно с таким числом рёбер описано в работе [7]. Однако в общем случае не обязательно, что граф с таким числом рёбер для заданных параметров существует. Например, оптимальные по числу рёбер n-вершинные графы с $k = \lambda = 1$ — это деревья с числом рёбер $n - 1$. Приводимые далее полные результаты были анонсированы в работе [8].

В [9] приводится список из 14 нерешённых задач теории графов. Задача № 11 следующая: какова наибольшая связность графа с n вершинами и m рёбрами? Решение было найдено Ф. Харари в работе [10]: наибольшая связность k графа с n вершинами и m рёбрами равна $\lceil 2m/n \rceil$ при $m \geqslant n - 1$. Достигается она на графах Харари $H_{k,n}$, в которых все вершины имеют степень k, если kn чётно, либо одна вершина имеет степень $k + 1$, а все остальные вершины имеют степень k, если kn нечётно. Рассмотрим в некотором смысле обратную задачу: какое минимальное число вершин и рёбер

может быть в графе с заданной вершинной связностью k и рёберной связностью λ. Соответствующие значения непосредственно следуют из результатов работы [5].

Обозначим через $N_{k,\lambda}$ минимальное число вершин, которое может содержать граф G с заданной вершинной связностью k и рёберной связностью λ:

$$N_{k,\lambda} = \begin{cases} 2(\lambda+1)-k & \text{при } \lambda > k, \\ \lambda+1 & \text{при } \lambda = k. \end{cases} \tag{1}$$

Обозначим через $E_{k,\lambda}$ минимальное число рёбер, которое может содержать граф G с заданной вершинной связностью k и рёберной связностью λ:

$$E_{k,\lambda} = \begin{cases} \lambda^2 - k^2 + k + \lambda + \sigma & \text{при } \lambda > k, \\ \lambda(\lambda+1)/2 & \text{при } \lambda = k, \end{cases} \tag{2}$$

где

$$\sigma = \begin{cases} 0, & \text{если } \lceil (2k^2 - k\lambda - 2k)/2 \rceil \leqslant 0, \\ \lceil (2k^2 - k\lambda - 2k)/2 \rceil & \text{иначе.} \end{cases}$$

Заметим, что если нас интересует минимальное число вершин N_k, которое может содержать граф с заданной вершинной связностью k, то получаем $N_k = N_{k,k} = k+1$, что соответствует полному графу с числом вершин $k+1$. Аналогично для случая минимального числа вершин N_λ, которое может содержать граф с заданной рёберной связностью λ.

Если нас интересует минимальное число рёбер E_k, которое может содержать граф с заданной вершинной связностью k, то получаем $E_k = E_{k,k} = k(k+1)/2$, что соответствует полному графу с числом вершин $k+1$. Аналогично для случая минимального числа рёбер E_λ, которое может содержать граф с заданной рёберной связностью λ.

Добавим ещё один параметр к поиску и будем рассматривать такую задачу: какое минимальное число рёбер может быть в n-вершинном графе с заданной вершинной связностью k и рёберной связностью λ? Обозначим это значение через $E_{k,\lambda,n}$. Из условия Уитни имеем, что минимальная степень вершины в графе $\delta \geqslant \lambda$, поэтому справедлива очевидная оценка:

$$E_{k,\lambda,n} \geqslant \lceil \lambda n/2 \rceil.$$

В работе [7] удалось описать множество значений k и λ, при которых достигается указанная оценка. Задача поиска оптимальных графов с максимальными мерами связности представляет интерес не только с теоретической, но и с практической точки зрения, например для построения отказоустойчивых сетей [11, 12].

2. Основной результат

Точкой сочленения в связном графе называется вершина, удаление которой вместе со всеми инцидентными ей рёбрами приводит к несвязному графу. Если в графе есть точка сочленения, то число вершинной связности $k = 1$. Рассмотрим сначала частный случай, когда и число рёберной связности $\lambda = 1$. По формулам (1) и (2) получаем, что $N_{1,1} = 2$ и $E_{1,1} = 1$. Очевидно, что минимальное число рёбер среди графов с n вершинами с $k = 1$ и $\lambda = 1$ имеют деревья. Как известно, n-вершинное дерево содержит $n-1$ ребро.

Далее рассмотрим случай $k = 1$ и $\lambda > 1$. В этом случае $N_{1,\lambda}$ можно вычислить следующим образом:

$$N_{1,\lambda} = 2(\lambda+1) - k = 2\lambda + 1.$$

Теорема 4. Пусть $k = 1$, $\lambda > 1$. Тогда для всех $n \geqslant N_{k,\lambda} = 2\lambda + 1$ минимальное число рёбер $E_{1,\lambda,n}$, которое может иметь n-вершинный граф с заданными k и λ, равно $[(\lambda n + \lambda + 1)/2]$. Эта оценка является достижимой. Соответствующий оптимальный по числу рёбер n-вершинный граф с заданными k и λ существует и имеет следующий вид:

- если λ чётное или n нечётное, то в таком графе степень одной вершины (точки сочленения) равна 2λ, а остальных — λ;
- если λ нечётное и n чётное, то граф может иметь один из двух видов:
  - степень одной вершины (точки сочленения) равна $2\lambda + 1$, а остальных — λ;
  - степень одной вершины (точки сочленения) равна 2λ, степень ещё одной вершины — $\lambda + 1$, а остальных — λ.

Доказательство. Рассмотрим произвольный n-вершинный граф G с $k = 1$, $\lambda > 1$, $n \geqslant 2\lambda+1$. Пусть вершина v — точка сочленения графа G. Согласно неравенству Уитни, минимальная степень вершины в графе G не может быть меньше λ. Покажем, что степень $d(v)$ точки сочленения не может быть меньше 2λ. Предположим обратное: пусть $d(v) < 2\lambda$. Удаление вершины v из графа G приводит к несвязному графу, который состоит не менее чем из двух компонент связности. Обозначим через G_1 одну из этих компонент, через G_2 — оставшуюся часть. Так как $d(v) < 2\lambda$, то вершина v инцидентна менее чем λ вершинам либо из G_1, либо из G_2. Не ограничивая общности, можно считать, что вершина v инцидентна менее чем λ вершинам из G_1. Удалив эти рёбра из графа G, мы нарушим его связность, а это противоречит тому, что рёберная связность графа G равна λ. Следовательно, степень точки сочленения не может быть меньше 2λ.

Таким образом, оптимальный по числу рёбер n-вершинный граф G с $k = 1$, $\lambda > 1$ должен иметь по крайней мере одну вершину степени не менее 2λ (точку сочленения) и остальные вершины — со степенью не ниже λ, что даёт оценку числа рёбер

$$E_{1,\lambda,n} \geqslant \lambda(n + 1)/2.$$

Однако с учётом того, что количество вершин нечётной степени должно быть чётно, оценку можно уточнить для случая, когда λ нечётно, а n чётно. Рассмотрим подграфы графа G, порождённые вершинами G_1 и вершиной v, G_2 и вершиной v. Обозначим через n_1 и n_2 число вершин в этих подграфах. Очевидно, что $n = n_1 + n_2 - 1$. Так как n чётно, количество вершин в одном из этих подграфов чётное, а в другом нечётное. Следовательно, в подграфе с нечётным числом вершин все вершины не могут иметь нечётную степень и одна из вершин должна иметь степень на 1 больше, чем указано. Это может быть либо вершина v, которая имеет степень $\lambda+1$ (и соответственно $2\lambda+1$ в графе G), либо какая-то другая вершина, которая имеет степень $\lambda + 1$. В любом случае это даёт следующую оценку числа рёбер:

$$E_{1,\lambda,n} \geqslant (\lambda(n + 1) + 1)/2.$$

С учётом первого случая можно записать оценку в общем виде:

$$E_{1,\lambda,n} \geqslant [(\lambda n + \lambda + 1)/2].$$

Далее покажем, что эта оценка является достижимой: n-вершинный граф с $k = 1$, $\lambda > 1$ и числом рёбер $[(\lambda n + \lambda + 1)/2]$ существует для всех $n \geqslant N_{k,\lambda} = 2\lambda + 1$.

Предыдущие рассуждения предлагают и схему построения соответствующего оптимального графа: необходимо взять два графа H_1 и H_2 с количеством вершин n_1 и n_2

с рёберной связностью λ. Если n_1 и n_2 чётные, то графы H_1 и H_2 должны быть λ-регулярными. Если число вершин в одном из графов нечётно, то он должнен быть почти λ-регулярным с единственной вершиной степени $\lambda+1$. Далее в графах H_1 и H_2 выбираются произвольные вершины v_1 и v_2, графы соединяются путём отождествления выбранных вершин. Получаем n-вершинный граф G, $n = n_1 + n_2 - 1$.

Для доказательства того, что эта схема реализуема, заметим, что достаточно в качестве графов H_1 и H_2 взять графы Харари H_{λ,n_1} и H_{λ,n_2}, степени вершин в которых имеют в точности указанные значения. Как известно, граф Харари $H_{k,n}$ существует для $k > 1$ и $n \geqslant k+1$ [9]. ■

Рассмотрим более подробно схему построения искомых графов из доказательства теоремы 4 для каждого случая по отдельности с примерами и обсудим вопрос единственности построения:

1) Пусть λ чётное или n нечётное. Графы H_1 и H_2 — два λ-регулярных графа с рёберной связностью λ, которые соединяются одной общей вершиной (точкой сочленения). Её степень равна 2λ. При этом оба подграфа для случая, когда λ нечётное, должны содержать чётное число вершин.

2) Пусть λ нечётное и n чётное. Один из подграфов перестаёт быть λ-регулярным: степень одной из его вершин равна $\lambda+1$. За счёт выбора вершины для отождествления появляется два варианта. Если выбираются вершины степени λ, то точка сочленения будет иметь степень 2λ, одна вершина — степень $\lambda+1$, а остальные вершины — степени λ. Либо вершина, которая является общей (точкой сочленения) для подграфов, со стороны одного подграфа имеет степень $\lambda+1$, со стороны другого — степень λ, степени всех остальных вершин равны λ. Таким образом, степень точки сочленения равна $2\lambda+1$.

На рис. 1 показана реализация графа с минимальным числом рёбер при $n = 6$, $\lambda = 2$, $k = 1$. Количество рёбер в этом графе равно $[(\lambda n+\lambda+1)/2] = [(12+2+1)/2] = 7$.

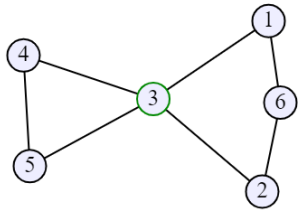

Рис. 1. Пример графа, удовлетворяющего первому случаю ($\lambda = 2$, $k = 1$, $n = 6$)

На рис. 2 показаны две реализации графа с минимальным числом рёбер при $n = 8$, $\lambda = 3$, $k = 1$. В первом случае точка сочленения (вершина 3) имеет степень $2\lambda = 6$, степень вершины 8 равна $4 = \lambda + 1$, а степени всех остальных вершин равны $\lambda = 3$. Во втором случае степени всех вершин, кроме точки сочленения, равны $\lambda = 3$. Степень точки сочленения со стороны левого подграфа равна $\lambda = 3$, а со стороны правого — $\lambda + 1 = 4$. Таким образом, степень точки сочленения равна $2\lambda + 1 = 7$. Очевидно, что графы неизоморфны. Количество рёбер в этих графах равно $[(\lambda n + \lambda + 1)/2] = [(24 + 3 + 1)/2] = 14$.

Последний пример показывает, что есть два неизоморфных 8-вершинных графа, которые имеют точку сочленения и рёберную связность $\lambda = 3$, с минимальным возможным числом рёбер. Формулировка теоремы указывает, что при нечётном $\lambda > 1$ и чётном $n \geqslant 2\lambda + 1$ существует как минимум два неизоморфных оптимальных

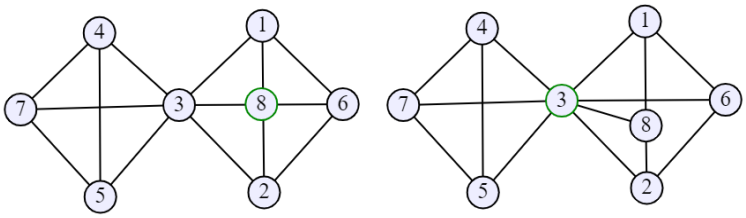

Рис. 2. Примеры графов, удовлетворяющих второму случаю ($\lambda = 3$, $k = 1$, $n = 8$)

n-вершинных графа с заданными значениями k и λ. Однако из доказательства следует, что в общем случае таких оптимальных графов может быть больше: вместо графа Харари можно взять любой k-связный граф с таким же числом рёбер либо графы Харари с разным числом вершин. Например, для случая $n = 7$, $\lambda = 2$ будет также два неизоморфных оптимальных графа. Действительно, имеем два способа выбора графов для соединения: два цикла с числом вершин 4 либо цикл с числом вершин 3 и цикл с числом вершин 5.

Выбор вершин для отождествления также может привести к неизоморфным оптимальным графам. Проведён эксперимент по вычислению количества неизоморфных оптимальных графов с точками сочленения и заданной рёберной связностью, его результаты приведены в таблице.

Количество оптимальных n-вершинных графов с точкой сочленения и рёберной связностью λ

n	λ				
	1	2	3	4	5
3	1	–	–	–	–
4	2	–	–	–	–
5	3	–	–	–	–
6	6	1	–	–	–
7	11	2	1	–	–
8	23	2	2	–	–
9	47	3	2	1	–
10	103	3	19	1	–
11	235	4	11	4	1

Как уже отмечалось, при $\lambda = 1$ оптимальными по числу рёбер графами являются деревья, значения в столбце 2 согласуются с известными данными [13]. Для $n < 10$ значения согласуются с данными сайта «Мир графов» [14]. Прочерки в таблице означают, что при соответствующих значениях n и λ подходящих графов не существует.

ЛИТЕРАТУРА

1. *Харари Ф.* Теория графов. М.: Мир, 1973.
2. *Богомолов А. М., Салий В. Н.* Алгебраические основы теории дискретных систем. М.: Наука, 1997.
3. *Whitney H.* Congruent graphs and the connectivity of graphs // Amer. J. Math. 1932. V. 54. Iss. 1. P. 150–168.
4. *Chartrand G. and Harary F.* Graphs with prescribed connectivities // Theory of Graphs. N.Y.: Academic Press, 1968. P. 61–63.
5. *Теребин Б. А., Абросимов М. Б.* Оптимальные реализации графов с заданными мерами связности // Матем. заметки. 2023. Т. 113. № 3. С. 323–331.

6. *Fulkerson D. R. and Shapley L. S.* Minimal k-arc-connected graphs // Networks. V. 1. No. 1. P. 91–98.
7. *Теребин Б. А., Абросимов М. Б.* Об одном семействе оптимальных графов с заданными мерами связности // Прикладная дискретная математика. Приложение. 2022. № 15. С. 116–119.
8. *Теребин Б. А., Абросимов М. Б.* О графах с заданной рёберной связностью, точками сочленения и минимальным числом рёбер // Материалы XIV Междунар. семинара «Дискретная математика и её приложения» имени академика О. Б. Лупанова (Москва, МГУ, 20–25 июня 2022 г.) / под ред. В. В. Кочергина. М.: ИПМ им. Келдыша, 2022. С. 200–203.
9. *Берж К. Ж.* Теория графов и её применения. М.: ИЛ, 1962. 323 с.
10. *Harary F.* The maximum connectivity of a graph // Proc. NAS USA. 1962. V. 48. P. 1142–1146.
11. *Steiglitz K., Weiner P., and Kleitman D.* The design of minimum-cost survivable networks // IEEE Trans. Circuit Theory. 1969. V. 16. No. 4. P. 455–460.
12. *Jafarpour M., Shekaramiz M., Javan A., and Moeini A.* Building graphs with maximum connectivity // Proc. IETS. Orem, UT, USA, 2020. P. 1–5.
13. `oeis.org/A000055` — Number of trees with n unlabeled nodes. 2025.
14. `graphworld.ru` — Мир графов. 2025.

REFERENCES

1. *Harary F.* Graph Theory. N.Y., Addison-Wesley, 1969. 274 p.
2. *Bogomolov A. M. and Saliy V. N.* Algebraicheskie osnovy teorii diskretnykh sistem [Algebraic foundations of the theory of discrete systems]. Moscow, Nauka, 1997. (in Russian)
3. *Whitney H.* Congruent graphs and the connectivity of graphs. // Amer. J. Math. 1932. V. 54. Iss. 1. P. 150–168.
4. *Chartrand G. and Harary F.* Graphs with prescribed connectivities. Theory of Graphs. N.Y., Academic Press, 1968, pp. 61–63.
5. *Terebin B. A. and Abrosimov M. B.* Optimal graphs with prescribed connectivities. Math. Notes, 2023, vol. 113, iss. 3, pp. 319–326.
6. *Fulkerson D. R. and Shapley L. S.* Minimal k-arc-connected graphs. Networks, vol. 1, no. 1, pp. 91–98.
7. *Terebin B. A. and Abrosimov M. B.* Ob odnom semeystve optimal'nykh grafov s zadannymi merami svyaznosti [One family of optimal graphs with prescribed connectivities]. Prikladnaya Diskretnaya Matematika. Prilozhenie, 2022, no. 15, pp. 116–119. (in Russian)
8. *Terebin B. A. and Abrosimov M. B.* O grafakh s zadannoy rebernoy svyaznost'yu, tochkami sochleneniya i minimal'nym chislom reber [On graphs with given edge connectivity, cut points and minimum number of edges]. Proc. XIV Intern. Seminar "Discrete Mathematics and its Applications" named after Academician O. B. Lupanov (MSU, June 20–25, 2022), Moscow, Keldysh Institute of Applied Mathematics, 2022, pp. 200–203. (in Russian)
9. *Berge C. J.* Theorie des graphes et ses applications. Dunod, 1958, 270 p. (in French)
10. *Harary F.* The maximum connectivity of a graph. Proc. NAS USA, 1962, vol. 48, pp. 1142–1146.
11. *Steiglitz K., Weiner P., and Kleitman D.* The design of minimum-cost survivable networks. IEEE Trans. Circuit Theory, 1969, vol. 16, no. 4, pp. 455–460.
12. *Jafarpour M., Shekaramiz M., Javan A., and Moeini A.* Building graphs with maximum connectivity. Proc. IETS, Orem, UT, USA, 2020, pp. 1–5.
13. `oeis.org/A000055` — Number of trees with n unlabeled nodes, 2025.
14. `graphworld.ru`. 2025.

МАТЕМАТИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ И ПРОГРАММИРОВАНИЯ

УДК 519.681:519.71 DOI 10.17223/20710410/70/5

ЭФФЕКТИВНЫЕ АЛГОРИТМЫ ПРОВЕРКИ ЭКВИВАЛЕНТНОСТИ ПРОПОЗИЦИОНАЛЬНЫХ ПРОГРАММ МИЛИ НА УРАВНОВЕШЕННЫХ ШКАЛАХ[1]

В. В. Подымов

МГУ имени М. В. Ломоносова, г. Москва, Россия

E-mail: valdus@yandex.ru

Предлагается и рассматривается модель программ, называемая далее моделью пропозициональных программ Мили (ППМ) и представляющая собой небольшое синтаксическое обобщение модели дискретных преобразователей Глушкова — Летичевского с «осовремененной» семантикой, основанной на понятиях, использующихся в модели пропозициональных последовательных программ, введённой В. А. Захаровым (ППЗ). Предлагается подход к построению эффективных алгоритмов проверки эквивалентности ППМ, являющийся адаптацией известного подхода к проверке эквивалентности ППЗ, основанного на анализе графа совместных вычислений программ. Демонстрируется применение этого подхода для получения эффективных алгоритмов проверки эквивалентности ППМ для некоторых видов семантик прикладного характера.

Ключевые слова: *проблема эквивалентности, проверка эквивалентности, модели программ, дискретные преобразователи, пропозициональные последовательные программы.*

EFFICIENT EQUIVALENCE-CHECKING ALGORITHMS FOR PROPOSITIONAL MEALY PROGRAMS OVER BALANCED FRAMES

V. V. Podymov

Lomonosov Moscow State University, Moscow, Russia

We propose and investigate propositional Mealy programs (PMPs), a model that is a slight syntactic generalization of the discrete processor model studied by V. M. Glushkov and A. A. Letichevsky. PMPs employ a “modernized” semantics based on notions used in the model of propositional sequential programs proposed by V. A. Zakharov (PSPZs). A technique for constructing efficient equivalence checking algorithms for PMPs is proposed, adapting a known technique for PSPZs based on analysis of a graph of consistent program computations. Efficient PMP equivalence-checking algorithms based on the proposed technique are obtained for some kinds of applied semantics.

[1]Исследования поддержаны Московским центром фундаментальной и прикладной математики МГУ имени М. В. Ломоносова по соглашению № 075-15-2025-345.

Keywords: *equivalence problem, equivalence checking, program models, discrete processors, propositional sequential programs.*

Введение

Данная работа посвящена исследованию проблемы эквивалентности программ: для двух заданных произвольных программ выяснить, имеют ли они одинаковое поведение. Из теоремы Райса — Успенского [1], констатирующей неразрешимость любого нетривиального свойства частично рекурсивных функций, и алгоритмической полноты этого класса функций [2] следует неразрешимость проблемы эквивалентности для любого достаточно выразительного (алгоритмически полного) класса программ. В связи с этим проблема эквивалентности исследуется, в числе прочего, для моделей программ с упрощённой семантикой, позволяющей избежать такой неразрешимости, с тем чтобы использовать решение этой проблемы в модели в качестве достаточного условия эквивалентности программ. Среди таких моделей нас интересуют дискретные преобразователи Глушкова — Летичевского (ДПГЛ) [3] и пропозициональные последовательные программы Захарова [4].

Общие черты моделей ДПГЛ и ППЗ таковы. Программа разделена на синтаксическую и семантическую части. Вычисление программы представляет собой взаимодействие этих двух частей, согласно которому выстраивается последовательность состояний управления и состояний данных программы, начиная с заданных входных значений и пока не будет достигнуто заданное выходное значение состояния управления либо до бесконечности. Результатом конечного вычисления объявляется последнее состояние данных. Построение вычисления основывается на операторах и логических условиях — символах, обозначающих соответственно способы изменения состояния данных программы и способы выбора следующего состояния управления в зависимости от текущих состояний управления и данных. Способ преобразования состояний данных операторами и выбор логических условий при продолжении вычисления задаются семантической частью. Выбор оператора, выполняющегося при продолжении вычисления, задаётся синтаксической частью.

Основные результаты, относящиеся к исследованию проблемы эквивалентности ДПГЛ [3, 5, 6], представляют собой разделение вариантов этой проблемы на разрешимые и неразрешимые. Но ввиду практической значимости алгоритмов проверки эквивалентности (подробнее о ней см., например, в [7]) интерес представляет не только разрешимость как таковая, но и эффективные решающие алгоритмы — полиномиальные достаточно низкой сложности. В [4] введена модель ППЗ, для неё предложена техника проектирования таких алгоритмов — техника совместных вычислений — и в конце введения коротко, без деталей и доказательств, отмечается, что модель ППЗ является обобщением модели ДПГЛ. В последующих работах, посвящённых проблеме эквивалентности ППЗ [7–21] (включая труды, упомянутые в списках литературы этих работ), это соотношение между моделями не обсуждается.

В данной работе обращается особое внимание на то, что пока ещё строго не установлено, имеет ли место включение между структурой или выразительными возможностями моделей ППЗ и ДПГЛ. Синтаксис этих моделей несравним: в модели ДПГЛ оператор обязан выполняться на каждом шаге вычисления, а в модели ППЗ может и не выполняться на последнем шаге; в модели ППЗ выполняющийся оператор однозначно задаётся следующим состоянием управления, а в модели ДПГЛ он может зависеть также и от выбора логического условия, аналогично тому, как выходной символ автомата Мура зависит только от состояния, тогда как в автомате Мили может

зависеть и от выбора входного символа [22]. При этом семантика ПППЗ, хотя и более «современна» в выборе терминологии по сравнению с семантикой ДПГЛ, но в конечном итоге не кажется более широкой, что коротко отмечается в том числе и в [23], хотя это сравнение требует более развёрнутого обсуждения.

В связи с обозначенным соотношением синтаксиса и семантики ДПГЛ и ПППЗ результаты, полученные для этих моделей, вообще говоря, следует считать независимыми, пока не будет строго установлена связь между этими моделями. В данной работе выполнен первый этап установления этой связи: предложена модель пропозициональных программ Мили, совмещающая синтаксис ДПГЛ и семантику ПППЗ, и к ней, в числе прочего, адаптированы результаты работы [4]: техника совместных вычислений и получающиеся с её помощью эффективные алгоритмы проверки эквивалентности для некоторых видов семантик прикладного характера. Применение предложенной техники, то есть эти виды семантик и детали соответствующих алгоритмов, описано в п. 7. Кроме того, получены некоторые побочные результаты, связанные с расширением и структурированием техники совместных вычислений, они обсуждаются в заключении.

Работа имеет следующую структуру. В п. 1 даются используемые понятия и обозначения общего характера. В п. 2 вводятся синтаксис и семантика ППМ, ставится рассматриваемая проблема эквивалентности и обсуждаются специальные понятия и факты, относящиеся к ППМ и требующиеся для формулировки результатов. В п. 3 приводится графовая конструкция, описывающая синхронное выполнение двух ППМ — граф совместных вычислений. В п. 4 вводится понятие критериальной системы, предназначенное для оценки каждой пары состояний данных ППМ характеристикой «удалённости» этих состояний друг от друга. В п. 5 обсуждается графовая конструкция, по сути представляющая собой граф совместных вычислений, снабжённый информацией об «удалённости» состояний данных описываемых пар вычислений, — критериальный граф. В п. 6 приводится алгоритм проверки эквивалентности ППМ, состоящий в обходе критериального графа и параметризованный выбором семантики и критериальной системы, с обоснованием корректности и оценкой сложности. Наконец, в п. 7 показывается, как критериальную систему из [4] можно переформулировать в виде критериальной системы данной работы, и обсуждаются алгоритмы проверки эквивалентности ППМ невысокой сложности, основанные на алгоритме п. 6 и критериальных системах из [4].

Некоторые утверждения в данной работе представляют собой адаптацию утверждений из [4] и смежные не очень сложно обосновываемые свойства рассматриваемых понятий. Такие утверждения, строго говоря, являются новыми и требуют обоснования, но эта новизна в основном техническая, а не содержательная, и эти утверждения озаглавлены словом «Утверждение». Утверждения, полагающиеся существенно новыми и нетривиальными, озаглавлены словами «Лемма» и «Теорема».

1. Общие понятия и обозначения

В связи с обилием областей, из которых далее обширно используются понятия и результаты, приведём названия этих понятий, кроме самых общеизвестных, со ссылками на работы, в которых можно их найти. Здесь же изложим сопутствующие обозначения, как взятые из упомянутых работ, так и отступающие от них.

Используемые понятия теории формальных языков [24]: алфавит; буква (символ) алфавита; слово в заданном алфавите; пустое слово; длина слова. Обозначения: λ —

пустое слово; Σ^* — множество всех слов в алфавите Σ; $|S|$ — размер множества S и длина последовательности S, в том числе длина слова.

Используемые понятия теории графов [25]: ориентированный граф с петлями и кратными дугами (далее — орграф); вершина и дуга орграфа; вершина, из которой исходит дуга и в которую заходит дуга; полный орграф; подграф; орграф, обратный к G; путь; длина пути; простой цикл; кратчайший путь от одной вершины до другой; достижимость одной вершины из другой. Рассмотрим также пути с вершинами заданного множества без упоминания графа, имея в виду пути в полном орграфе с этими вершинами. Про путь будем говорить, что он исходит из первой своей вершины и если он конечен, то он ведёт в последнюю свою вершину. Обозначения: $v_1 \to v_2$ — дуга (v_1, v_2) и часть пути $(v_1, (v_1, v_2), v_2)$; $\rho(i) = v_i$ и $\rho|^n = (v_0 \to v_1 \to \cdots \to v_n)$ для пути $\rho = v_0 \to v_1 \to \ldots$ Будем называть отрезком пути $v_0 \to v_1 \to \ldots$ всякую часть $v_i \to \cdots \to v_j$ этого пути и такой отрезок начальным, если $i = 0$. Как и в [25], рассмотрим также орграфы с помеченными вершинами и дугами, считая, что метки сохраняются при преобразовании графов и рассмотрении и преобразовании путей, и использовать в примерах понятие изоморфизма таких графов. Дугу $v_1 \to v_2$, помеченную значением x, будем записывать как $v_1 \xrightarrow{x} v_2$; кратные дуги $v_1 \to v_2$ с разными метками $x_1, \ldots, x_k$ — как одну помеченную дугу $v_1 \xrightarrow{x_1,\ldots,x_k} v_2$.

Будем использовать также понятия сужения функции $f : X \to Y$ на множество $Z \subseteq X$ [26] и частично определённой функции [2] и следующие обозначения: $f|_Z$ — сужение функции f на Z; $\perp$ — значение неопределённости, не входящее ни в одно из рассматриваемых множеств, кроме случаев, когда это сказано явно; $f : X \to Y \cup \{\perp\}$ — частично определённая функция, в которой $f(x) = \perp$ означает, что значение $f(x)$ не определено; $2^X = \{Y : Y \subseteq X\}$; $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$.

При обсуждении алгоритмов следуем терминологии и рекомендациям [27] и используем известные структуры данных: одномерный и двумерный массивы (далее — соответственно векторы и матрицы); односвязный список; список смежности ориентированного графа. Под сложностью алгоритма понимаем его сложность по времени в худшем случае в модели RAM [27, разд. 2.2]. Это означает, в частности, что при подсчёте сложности полагаем, что в распоряжении алгоритма есть любое необходимое количество пронумерованных ячеек памяти, способных хранить любые целые числа, и единица сложности отвечает выполнению любой простейшей команды: переход к заданной команде, безусловный или условный с проверкой значения в заданной ячейке и сравнениями $(=, \neq, <, \leqslant, >, \geqslant)$ значений в ячейках; вычисление в заданной ячейке суммы, разности, произведения, частного или остатка от деления значений заданных ячеек; копирование значения из одной ячейки в другую. Для алгоритма $\mathcal{A}$ записью $\mathcal{A}(x)$ будем обозначать результат выполнения $\mathcal{A}$ на входе x.

Используемые понятия общей алгебры [28, 29]: моноид; образующие (порождающие) элементы; определяющие соотношения; подмоноид; конечно порождённый моноид; свободный моноид; свободный коммутативный моноид; частично коммутативный моноид; прямое произведение моноидов; гомоморфизм моноида A на моноид B (сюръективное отображение элементов A в элементы B, сохраняющее нейтральный элемент и операцию). Для краткости будем называть моноид с множеством образующих X просто X-моноидом. Обозначения: $\mathcal{M} = (M, \varepsilon, \circ)$ — моноид с множеством элементов M, нейтральным элементом ε и операцией $\circ$; $m \in \mathcal{M}$ — синоним записи $m \in M$; $\mathcal{M}(a_1 \ldots a_k) = a_1 \circ \cdots \circ a_k$, где $a_1, \ldots, a_k \in \mathcal{M}$; $m \circ \mathcal{U} = \{m \circ x : x \in \mathcal{U}\}$; $\mathcal{U} \circ m = \{x \circ m : x \in \mathcal{U}\}$; $\mathcal{M}_1 \times \mathcal{M}_2$ — прямое произведение моноидов $\mathcal{M}_1$ и $\mathcal{M}_2$.

2. Модель программ

2.1. Синтаксис

Символами $\mathfrak{A}$ и $\mathfrak{C}$ будем обозначать конечные непустые множества *операторов* и *логических условий* соответственно; эти два множества считаются заданными. Слова в алфавите $\mathfrak{A}$ будем называть (операторными) *цепочками*.

Пропозициональной программой Мили над $\mathfrak{A}$ и $\mathfrak{C}$ (далее — $(\mathfrak{A},\mathfrak{C})$-*программой* и просто *программой*) будем называть систему $\pi = (S, en, EX, T)$, где:

— S — непустое множество *состояний*;
— $en \in S$ — *вход*;
— $EX \subseteq S$ — множество *выходов*;
— $T : (S \setminus EX) \times \mathfrak{C} \to (\mathfrak{A} \times S) \cup \{\perp\}$ — частично определённая *функция переходов*.

Будем также использовать следующие обозначения: если $T(s,c) = (a,r)$, то $T^{\mathfrak{A}}(s,c) = a$ и $T^{S}(s,c) = r$. *Переходом* программы π будем называть четвёрку (s,c,a,r), для которой верно $T(s,c) = (a,r)$. Будем считать, что программа π представляет собой размеченный орграф, в котором S — множество вершин, «вход» и «выход» — метки вершин и каждый переход (s,c,a,r) представляет собой помеченную дугу $s \xrightarrow{c/a} r$.

Пример 1. На рис. 1 показаны следующие пропозициональные программы Мили над множествами $\mathfrak{A} = \{a,b\}$ и $\mathfrak{C} = \{c_1,c_0\}$: $\pi_1 = (\{s_1,s_2,s_3,s_4\}, s_1, \{s_3\}, T_1)$, $\pi_2 = (\{r_1,r_2,r_3,r_4\}, r_1, \{r_3,r_4\}, T_2)$, таблицы значений функций переходов T_1 и T_2 приведены на рис. 2. Здесь и далее вход помечается символом $*$, а выход — двойным контуром. Содержательное понимание элементов синтаксиса пропозициональных программ Мили совпадает с пониманием соответствующих элементов синтаксиса дискретных преобразователей Глушкова — Летичевского и может быть почерпнуто из [3, разд. 2]. В данной работе ограничимся небольшим примером для иллюстрации основных понятий и результатов. На рис. 3 приведён фрагмент кода на языке C++ [30], которому отвечает программа π_1, если буквами a и b обозначены соответственно присваивания «x = x + 1» и «y = y + 1», а буквами c_1 и c_0 — множества состояний данных, в которых соответственно истинно и ложно условие «y < 1».

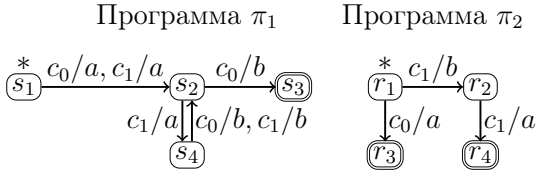

Рис. 1. Пропозициональные программы Мили (пример 1)

$T_1(\mathfrak{s},\mathfrak{c})$		$\mathfrak{c}$	
		c_0	c_1
$\mathfrak{s}$	s_1	a, s_2	a, s_2
	s_2	b, s_3	a, s_4
	s_4	b, s_2	b, s_2

$T_2(\mathfrak{s},\mathfrak{c})$		$\mathfrak{c}$	
		c_0	c_1
$\mathfrak{s}$	r_1	a, r_3	b, r_2
	r_2	$\perp$	a, r_4

Рис. 2. Таблицы значений функций переходов программ (пример 1)

```
x = x + 1;
while(y < 1) {
  x = x + 1;
  y = y + 1;
}
y = y + 1;
```

Рис. 3. Фрагмент кода на языке C++ (пример 1)

Путь в программе и в других графах будем называть *входным*, если он исходит из входа, и *выходным*, если он конечен и ведёт в выход. *Цепочкой пути* $\rho = (s_0 \xrightarrow{c_1/a_1} s_1 \xrightarrow{c_2/a_2} \dots)$ будем называть цепочку $a_1 a_2 \dots$ (для бесконечного пути — бесконечную последовательность операторов).

Программу будем называть *конечной*, если множество её состояний конечно, и *полной*, если её функция переходов всюду определена. *Размером* $|\pi|$ конечной программы $\pi = (S, en, EX, T)$ будем называть число $|S|$.

2.2. Семантика

Детерминированной динамической шкалой над $\mathfrak{A}$ (далее — $\mathfrak{A}$-*шкалой* и просто *шкалой*) называется система $\mathcal{F} = (D, d^0, \circ)$, где:

— D — непустое множество *состояний*;
— $d^0 \in D$ — *вход*;
— $\circ : D \times \mathfrak{A} \to D$ — *операция шкалы*.

Переходом шкалы $\mathcal{F}$ будем называть тройку (d, a, e), удовлетворяющую равенству $d \circ a = e$. Считаем, что шкала $\mathcal{F}$ представляет собой размеченный орграф, в котором D — множество вершин, «вход» — метка вершин и переход (d, a, e) представляет собой помеченную дугу $d \xrightarrow{a} e$. *Цепочкой пути* $\rho = (d_0 \xrightarrow{a_1} d_1 \xrightarrow{a_2} \dots)$ будем называть цепочку $a_1 a_2 \dots$ (для бесконечного пути — бесконечную последовательность операторов). Записью $\mathcal{F}(d, h)$ для $d \in D$ и $h \in \mathfrak{A}^*$ будем обозначать последнее состояние пути в $\mathcal{F}$, исходящего из d и имеющего цепочку h, и записью $\mathcal{F}(h)$ — состояние $\mathcal{F}(d^0, h)$.

Пример 2. На рис. 4 представлен фрагмент шкалы $\mathcal{F}^{x_0,y_0} = (\{x_0, x_0+1, x_0+2, \dots\} \times \{y_0, y_0+1, y_0+2, \dots\}, (x_0, y_0), \circ)$ над $\mathfrak{A} = \{a, b\}$, где $x_0, y_0 \in \mathbb{Z}$, $(x, y) \circ a = (x+1, y)$ и $(x, y) \circ b = (x, y+1)$. Этой шкалой определяется семантика операторов из примера 1 для (неограниченных) целочисленных переменных x, y, имеющих значения x_0 и y_0 соответственно в начале выполнения программы. Заметим, что для всех $x_0, y_0 \in \mathbb{Z}$ шкалы $\mathcal{F}^{x_0,y_0}$ изоморфны как размеченные графы. Поэтому далее в примерах в основном используем шкалу $\mathcal{F}^{0,0}$.

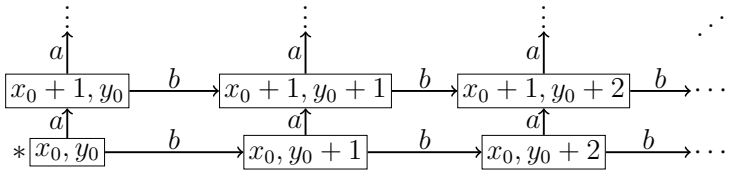

Рис. 4. Фрагмент детерминированной динамической шкалы (пример 2)

Графом $\mathcal{F}$-*вычислений* программы $\pi = (S, en, EX, T)$ назовём программу $\pi \oplus \mathcal{F} = (S \times D, (en, d^0), EX \times D, \mathcal{T})$, в которой функция переходов $\mathcal{T}$ задаётся так: если $T(s, c) = \perp$, то $\mathcal{T}((s, d), c) = \perp$, иначе $\mathcal{T}((s, d), c) = (T^{\mathfrak{A}}(s, c), (T^S(s, c), d \circ T^{\mathfrak{A}}(s, c)))$. Вершины графа $\pi \oplus \mathcal{F}$ и пути в нём будем называть соответственно $\mathcal{F}$-*конфигурациями* и $\mathcal{F}$-*трассами* программы π, элементы s и d конфигурации (s, d) — соответственно *состоянием управления* и *состоянием данных* этой конфигурации. Путь в программе и путь в шкале будем называть *парными*, если их цепочки одинаковы. Для парных путей $\rho_1 = (s_0 \xrightarrow{c_1/a_1} s_1 \xrightarrow{c_2/a_2} \dots)$ и $\rho_2 = (d_0 \xrightarrow{a_1} d_1 \xrightarrow{a_2} \dots)$ одинаковой длины в программе и шкале соответственно записью $\rho_1 \oplus \rho_2$ обозначаем путь $(s_0, d_0) \xrightarrow{c_1/a_1} (s_1, d_1) \xrightarrow{c_2/a_2} \dots$; пути ρ_1, ρ_2 называем соответственно *путём управления* и *путём данных* пути $\rho_1 \oplus \rho_2$.

Пример 3. На рис. 5 приведён фрагмент графа вычислений программы π_1 из примера 1 на шкале $\mathcal{F}^{0,0}$ из примера 2. Входной путь $(s_1,(0,0)) \xrightarrow{c_1/a} (s_2,(1,0)) \xrightarrow{c_0/b} (s_3,(1,1))$ в этом графе можно представить в виде $\rho_1 \oplus \rho_2$, где путь управления $\rho_1 = \big(s_1 \xrightarrow{c_1/a} s_2 \xrightarrow{c_0/b} s_3\big)$ — это входной путь в π_1 и путь данных $\rho_2 = \big((0,0) \xrightarrow{a} (1,0) \xrightarrow{b} (1,1)\big)$ — это парный входной путь в $\mathcal{F}^{0,0}$.

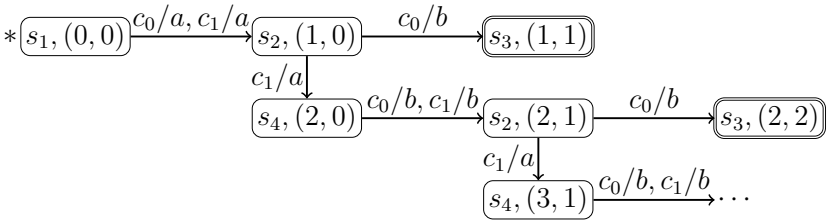

Рис. 5. Фрагмент графа вычислений (пример 3)

Утверждение 1. Для любых программы π и шкалы $\mathcal{F}$ входными путями в $\pi \oplus \mathcal{F}$ являются всевозможные пути $\rho_1 \oplus \rho_2$, где ρ_1 и ρ_2 — парные пути в π и $\mathcal{F}$, и только они.

Доказательство. Пусть $\pi = (S, en, EX, T)$ и $\mathcal{F} = (D, d^0, \circ)$. По устройству путей вида $\rho_1 \oplus \rho_2$ и графа $\pi \oplus \mathcal{F}$, во-первых, все такие пути начинаются со входа этого графа, и, во-вторых, множества переходов $(s,d) \xrightarrow{c/a} (r,e)$, исходящих из заданной конфигурации (s,d) в таких путях и в этом графе, равны: метка c/a произвольно выбирается среди меток переходов, исходящих из s в π; $r = T(s)$; $e = d \circ a$. ■

Детерминированной динамической моделью над $\mathfrak{A}$ и $\mathfrak{C}$ (далее — просто *моделью*) называется система $\mathcal{I} = (\mathcal{F}, L)$, где $\mathcal{F} = (D, d^0, \circ)$ — шкала и $L : D \to \mathfrak{C}$. Такую модель будем также называть *$\mathcal{F}$-моделью* и считать графом, получающимся из $\mathcal{F}$ пометкой каждой вершины d значением $L(d)$. Будем называть *$\mathcal{I}$-трассой*, а также трассой, *реализующейся в* $\mathcal{I}$, $\mathcal{F}$-трассу, для каждого перехода $(s,d) \xrightarrow{c/a} \sigma$ которой верно $c = L(d)$. Будем называть $\mathcal{I}$-трассу τ программы *полной*, если она является самой длинной среди всех $\mathcal{I}$-трасс этой программы, исходящих из $\tau(0)$. Полную входную $\mathcal{I}$-трассу программы будем называть *$\mathcal{I}$-вычислением* этой программы.

Пример 4. На рис. 6 приведён фрагмент модели $\mathcal{I}_1$ над $\mathfrak{A} = \{a, b\}$ и $\mathfrak{C} = \{c_0, c_1\}$, содержащей шкалу $\mathcal{F}^{0,0}$ из примера 2 и разметку её состояний логическими условиями согласно содержательной трактовке из примера 1. Рассмотрим также $\mathcal{F}^{0,0}$-модель $\mathcal{I}_2$, в которой каждое состояние помечено условием c_1, и программы π_1 и π_2 из примера 1. $\mathcal{I}_1$-вычисление τ_1 программы π_1 устроено так: $(s_1,(0,0)) \xrightarrow{c_1/a} (s_2,(1,0)) \xrightarrow{c_1/a} (s_4,(2,0)) \xrightarrow{c_1/b} (s_2,(2,1)) \xrightarrow{c_0/b} (s_3,(2,2))$. $\mathcal{I}_2$-вычисление τ_2 программы π_1 бесконечно и начинается так: $(s_1,(0,0)) \xrightarrow{c_1/a} (s_2,(1,0)) \xrightarrow{c_1/a} (s_4,(2,0)) \xrightarrow{c_1/b} (s_2,(2,1)) \xrightarrow{c_1/a} (s_4,(3,1)) \xrightarrow{c_1/b} \ldots$ $\mathcal{I}_1$-вычисление τ_3 программы π_2 устроено так: $(r_1,(0,0)) \xrightarrow{c_1/b} (r_2,(0,1))$.

Утверждение 2. Для любой программы π и любой модели $\mathcal{I}$ существует ровно одно $\mathcal{I}$-вычисление программы π.

Доказательство. Пусть $\mathcal{I} = (\mathcal{F}, L)$.

С у щ е с т в о в а н и е . Входной путь длины 0 в графе $\pi \oplus \mathcal{F}$ существует и является входной $\mathcal{I}$-трассой. Значит, существует и входная $\mathcal{I}$-трасса наибольшей длины.

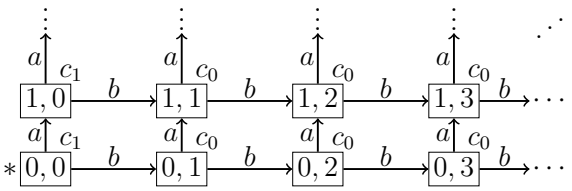

Рис. 6. Фрагмент детерминированной динамической модели (пример 4)

Единственность. Достаточно показать единственность входной $\mathcal{I}$-трассы τ программы π заданной длины. Для этого заметим, что 1) первая конфигурация каждой $\mathcal{F}$-трассы задана однозначно и 2) для каждого перехода $(s,d) \xrightarrow{c/a} (r,e)$ значениями s и d однозначно задаются остальные значения: $c = L(d)$, в π содержится не более одного перехода, исходящего из s и помеченного условием c, и этим переходом однозначно задаются a, r и $e = \mathcal{F}(d,a)$. ■

Далее будем без отсылки к утверждению 2 использовать понятие $\mathcal{I}$-вычисления программы для модели $\mathcal{I}$, имея в виду, что такое вычисление существует и единственно. Будем называть *итогом* конечной трассы состояние данных её последней конфигурации, *результатом* выходной трассы — её итог, а результатами трасс, не являющихся выходными, — значение $\perp$. Результат $\mathcal{I}$-вычисления программы π обозначим $\mathcal{I}(\pi)$. Для шкалы $\mathcal{F}$ будем называть *$\mathcal{F}$-вычислением* $\mathcal{I}$-вычисление для любой $\mathcal{F}$-модели $\mathcal{I}$.

Пример 5. Результаты вычислений τ_1, τ_2 и τ_3, приведённых в примере 4, равны соответственно $(2,2)$, $\perp$ и $\perp$.

2.3. Эквивалентность

Программы π_1, π_2 будем называть *эквивалентными в модели $\mathcal{I}$*, а также *$\mathcal{I}$-эквивалентными*, если $\mathcal{I}(\pi_1) = \mathcal{I}(\pi_2)$, и *эквивалентными на шкале $\mathcal{F}$*, а также *$\mathcal{F}$-эквивалентными*, если они эквивалентны в каждой $\mathcal{F}$-модели. Будем обозначать $\mathcal{F}$-эквивалентность программ π_1 и π_2 как $\pi_1 \sim_{\mathcal{F}} \pi_2$. *Проблема эквивалентности программ на шкале $\mathcal{F}$* состоит в том, чтобы для заданных произвольных конечных программ π_1, π_2 проверить соотношение $\pi_1 \sim_{\mathcal{F}} \pi_2$. Будем называть программы *сильно эквивалентными*, если они эквивалентны в любой модели (а значит, и на любой шкале).

Пример 6. Эквивалентность программ на шкале означает, что результаты вычислений этих программ обязательно равны, если смысл логических условий неизвестен, а относительно операторов известны только свойства, задаваемые этой шкалой. В частности, шкала $\mathcal{F}^{0,0}$ из примера 2 отвечает свойству перестановочности (коммутативности) операторов a и b: итог трассы не зависит от порядка выполнения операторов. Этим свойством обладают присваивания, которым сопоставлены буквы a и b в примере 1. Для программ π_1 и π_2 из примера 1 верно $\mathcal{I}_1(\pi_1) \neq \mathcal{I}_1(\pi_2)$ (см. примеры 4 и 5), а значит, $\pi_1 \nsim_{\mathcal{F}^{0,0}} \pi_2$.

2.4. Вспомогательные понятия и свойства

Будем считать заданной шкалу $\mathcal{F} = (D, d^0, \circ)$ и все утверждения сформулируем для произвольной такой шкалы.

Шкалу $\mathcal{F}$ будем называть *уравновешенной*, если для любых цепочек h, g из равенства $\mathcal{F}(h) = \mathcal{F}(g)$ следует $|h| = |g|$. Записями $\mathcal{R}_{\mathcal{F}}$ и $\mathcal{B}_{\mathcal{F}}$ обозначаем соответственно множества $\{\mathcal{F}(h) : h \in \mathfrak{A}^*\}$ и $\{(\mathcal{F}(h), \mathcal{F}(g)) : h, g \in \mathfrak{A}^*, |h| = |g|\}$.

Пример 7. Шкала $\mathcal{F}^{0,0}$ из примера 2 уравновешенна, так как для любых $k, m \in \in \mathbb{N}_0$ все слова h, для которых $\mathcal{F}^{0,0}(h) = (k,m)$, имеют одинаковую длину $(k+m)$.

Шкала $\mathcal{F} = (\{d\}, d, \circ)$ над $\mathfrak{A} = \{a, b\}$, в которой $d \circ a = d \circ b = d$, не уравновешенна: $\mathcal{F}(a) = \mathcal{F}(aa)$, но $|a| \neq |aa|$.

Пополнением программы $\pi = (S, en, EX, T)$ для значений $\mathrm{loop} \notin S$ и $a \in \mathfrak{A}$ будем называть программу $\pi^{\mathrm{loop},a} = (S \cup \{\mathrm{loop}\}, en, EX, T^{a,\mathrm{loop}})$, где $T^{\mathrm{loop},a}$ отличается от T только тем, что если $T(s,c) = \bot$ или $s = \mathrm{loop}$, то $T^{\mathrm{loop},a}(s,c) = (a, \mathrm{loop})$. Состояние s программы назовём *завершаемым*, если из него в этой программе достижим выход, иначе — *незавершаемым*. Весом $\|s\|_\pi$ состояния s программы π считаем длину кратчайшего пути из s в какой-либо выход в π (если такого пути нет, то вес бесконечен).

Пример 8. Пополнение $\pi_2^{\mathrm{loop},a}$ программы π_2 из примера 1 приведено на рис. 7.

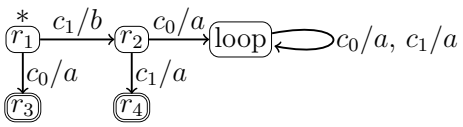

Рис. 7. Пополнение программы (пример 8)

Пример 9. Для программ π_1 из примера 1 и $\pi_2' = \pi_2^{\mathrm{loop},a}$ из примера 8 верно следующее: $\|s_3\|_{\pi_1} = \|r_3\|_{\pi_2'} = \|r_4\|_{\pi_2'} = 0$; $\|s_2\|_{\pi_1} = \|r_1\|_{\pi_2'} = \|r_2\|_{\pi_2'} = 1$; $\|s_1\|_{\pi_1} = \|s_4\|_{\pi_1} = 2$; $\|\mathrm{loop}\|_{\pi_2'} = \infty$. Состояние loop незавершаемо, остальные состояния завершаемы.

Назовём $\mathcal{F}$-трассы программ *совместными* (в том числе одну трассу — совместной), если существует $\mathcal{F}$-модель, в которой реализуются все эти трассы. *Произведением $\mathcal{F}$-трасс* $\tau_1 = ((s_0, d_0) \xrightarrow{c_1/a_1} (s_1, d_1) \xrightarrow{c_2/a_2} \dots)$ и $\tau_2 = ((r_0, e_0) \xrightarrow{\ell_1/b_1} (r_1, e_1) \xrightarrow{\ell_2/b_2} \dots)$ одинаковой длины будем называть путь $\tau_1 \otimes \tau_2 = (((s_0, r_0), (d_0, e_0)) \xrightarrow{(c_1,\ell_1)/(a_1,b_1)} ((s_1, r_1), (d_1, e_1)) \xrightarrow{(c_2,\ell_2)/(a_2,b_2)} \dots)$, такие трассы τ_1 и τ_2 будем называть соответственно *первой и второй проекциями* этого пути.

Пример 10. Рассмотрим модели $\mathcal{I}_1, \mathcal{I}_2$ и $\mathcal{F}^{0,0}$-трассы τ_1, τ_2, τ_3 из примера 4. Трассы τ_1 и τ_3 совместны, так как обе они реализуются в $\mathcal{I}_1$. Трассы τ_2 и τ_3 совместны, так как обе они реализуются в $\mathcal{I}_2$. Трассы τ_1 и τ_2 несовместны, так как если они обе реализуются в некоторой модели $\mathcal{I}$, то, согласно устройству τ_1, состояние $(2,1)$ в $\mathcal{I}$ должно быть помечено условием c_0, а согласно устройству τ_2, — условием c_1. Путь $\tau_1|^1 \otimes \tau_3$ имеет вид $((s_1, r_1), ((0,0),(0,0))) \xrightarrow{(c_1,c_1)/(a,b)} ((s_2, r_2), ((1,0),(0,1)))$.

Утверждение 3. Любые две программы $\mathcal{F}$-эквивалентны тогда и только тогда, когда результаты любых совместных $\mathcal{F}$-вычислений этих двух программ равны.

Доказательство. Напрямую следует из соответствующих определений. ■

Утверждение 4. Любая программа сильно эквивалентна любому своему пополнению.

Доказательство. Для любой модели $\mathcal{I}$ $\mathcal{I}$-вычисления программы и её пополнения отличаются только тем, что если вычисление программы конечно и ведёт не в выход, то вычисление её пополнения бесконечно. При этом результаты всех конечных $\mathcal{I}$-вычислений, оканчивающихся не в выходе, и бесконечных вычислений равны $\bot$. Значит, для любой модели $\mathcal{I}$ результаты $\mathcal{I}$-вычислений программы и её пополнения равны. ■

Утверждение 5. Любая конечная $\mathcal{F}$-трасса τ любой программы имеет итог $\mathcal{F}(d, h)$, где d — состояние данных конфигурации $\tau(0)$ и h — цепочка трассы τ.

Доказательство. Достаточно применить индукцию по длине трассы, заметив, что для любого перехода $(s, \mathcal{F}(d,h)) \xrightarrow{c/a} (r,e)$ трассы τ верно $e = \mathcal{F}(d, ha)$ (по определению трассы). ■

Утверждение 6. Итогом любой конечной входной $\mathcal{F}$-трассы τ любой программы является значение $\mathcal{F}(h)$, где h — цепочка трассы τ.

Доказательство. Следует из утверждения 5 и того, что состояние данных конфигурации $\tau(0)$ есть $\mathcal{F}(\lambda)$ (по соответствующим определениям). ■

Утверждение 7. Для любой вершины (ss, dd) произведения $\tau_1 \otimes \tau_2$ любых входных $\mathcal{F}$-трасс τ_1, τ_2 одинаковой длины любых программ верно $dd \in \mathcal{B}_{\mathcal{F}}$.

Доказательство. Достаточно показать, что утверждение верно для последней вершины (ss, dd) каждого конечного начального отрезка ρ пути $\tau_1 \otimes \tau_2$. Пусть путь ρ имеет длину n. По определению произведения трасс $\rho = \tau_1|^n \otimes \tau_2|^n$ и $dd = (d_1, d_2)$ для состояний данных d_1, d_2 конфигураций $\tau_1(n)$, $\tau_2(n)$. По утверждению 6 для некоторых цепочек h и g одинаковой длины n верно $d_1 = \mathcal{F}(h)$ и $d_2 = \mathcal{F}(g)$. Значит, $(d_1, d_2) \in \mathcal{B}_{\mathcal{F}}$ по определению этого множества. ■

Утверждение 8. Любой набор $\mathcal{F}$-трасс любых программ совместен в том и только в том случае, если для любых двух переходов $(s_1, d_1) \xrightarrow{c_1/a_1} \sigma_1$ и $(s_2, d_2) \xrightarrow{c_2/a_2} \sigma_2$ любых трасс этого набора из равенства $d_1 = d_2$ следует равенство $c_1 = c_2$.

Доказательство.

Необходимость. Если набор $\mathcal{F}$-трасс совместен, то существует модель $\mathcal{I} = (\mathcal{F}, L)$, в которой реализуются все трассы этого набора, и для любого состояния данных d и любых переходов $(s_1, d) \xrightarrow{c_1/a_1} \sigma_1$ и $(s_2, d) \xrightarrow{c_2/a_2} \sigma_2$ трасс этого набора $c_1 = L(d) = c_2$.

Достаточность. Пусть свойство переходов, сформулированное в условии, верно. Рассмотрим любую модель $\mathcal{I} = (\mathcal{F}, L)$, обладающую таким свойством: если в какой-либо из трасс содержится переход $(s, d) \xrightarrow{c/a} \sigma$, то $L(d) = c$. Согласно предполагаемому свойству переходов, для любого перехода $(r, d) \xrightarrow{\ell/b} \delta$ рассматриваемого набора трасс верно $\ell = c = L(d)$. Значит, интерпретация $\mathcal{I}$ задана корректно (существует). По определению все трассы рассматриваемого набора реализуются в $\mathcal{I}$. ■

Утверждение 9. Любые отрезки любых совместных $\mathcal{F}$-трасс совместны.

Доказательство. Следует из утверждения 8 и того, что все переходы отрезка трассы являются переходами этой трассы. ■

Утверждение 10. Любые $\mathcal{F}$-трассы любых программ совместны тогда и только тогда, когда совместна каждая пара конечных начальных отрезков этих трасс.

Доказательство.

Необходимость следует из утверждения 9.

Достаточность. Рассмотрим произвольный несовместный набор трасс. По утверждению 8 в этих трассах существуют переходы $t_1 = ((s_1, d_1) \xrightarrow{c_1/a_1} \sigma_1)$ и $t_2 = ((s_2, d_2) \xrightarrow{c_2/a_2} \sigma_2)$, для которых $d_1 = d_2$ и $c_1 \neq c_2$. Тогда существуют и конечные начальные отрезки τ_1', τ_2' некоторых трасс этого набора, содержащие переходы t_1 и t_2 соответственно. Трассы τ_1', τ_2' несовместны по утверждению 8. ■

3. Граф совместных вычислений

В данном пункте считаем заданными полные программы $\pi_1 = (S_1, en_1, EX_1, T_1)$ и $\pi_2 = (S_2, en_2, EX_2, T_2)$ и уравновешенную шкалу $\mathcal{F} = (D, d^0, \circ)$. Все утверждения формулируются для произвольных таких π_1, π_2 и $\mathcal{F}$.

Произведением программ π_1 и π_2 будем называть $(\mathfrak{A} \times \mathfrak{A}, \mathfrak{C} \times \mathfrak{C})$-программу $\pi_1 \otimes \pi_2 = (S_1 \times S_2, (en_1, en_2), (EX_1 \times S_2) \cup (S_1 \times EX_2), T)$, функция переходов которой задаётся так: если $T_1(s, c) \neq \perp$ и $T_2(r, \ell) \neq \perp$, то $T((s, r), (c, \ell)) = ((T_1^{\mathfrak{A}}(s, c), T_2^{\mathfrak{A}}(r, \ell)), (T_1^{S_1}(s, c), T_2^{S_2}(r, \ell)))$, иначе $T((s, r), (c, \ell)) = \perp$. *Произведением шкал* $\mathcal{F}_1 = (D_1, d_1^0, \circ_1)$ и $\mathcal{F}_2 = (D_2, d_2^0, \circ_2)$ называем $(\mathfrak{A} \times \mathfrak{A})$-шкалу $\mathcal{F}_1 \otimes \mathcal{F}_2 = (D_1 \times D_2, (d_1^0, d_2^0), \circ)$, в которой операция $\circ$ задаётся равенством $(d, e) \circ (a, b) = (d \circ_1 a, e \circ_2 b)$. Для $(\mathfrak{A} \times \mathfrak{A})$-шкалы $\mathcal{F}$ и цепочек $h = a_1 \dots a_k$ и $g = b_1 \dots b_k$, где $k \in \mathbb{N}_0$ и $a_1, \dots a_k, b_1, \dots, b_k \in \mathfrak{A}$, используем сокращения $\mathcal{F}(d, h_1, h_2) = \mathcal{F}(d, (a_1, b_1) \dots (a_k, b_k))$ и $\mathcal{F}(h_1, h_2) = \mathcal{F}(d^0, h_1, h_2)$. *Графом совместных вычислений* программ π_1 и π_2 на шкале $\mathcal{F}$ назовём подграф $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$ графа $(\pi_1 \otimes \pi_2) \oplus (\mathcal{F} \otimes \mathcal{F})$, содержащий все вершины и дуги, кроме дуг вида $((s, r), (d, e)) \xrightarrow{(c,\ell)/(a,b)} v$, в которых $d = e$ и $c \neq \ell$.

Пример 11. Рассмотрим программы π_1 и $\pi_2' = \pi_2^{\text{loop},a}$ и шкалу $\mathcal{F}^{0,0}$ из примеров 1, 2 и 8. Положим $\mathfrak{c}_{ij} = (c_i, c_j)$, $\mathfrak{a}_{xy} = (x, y)$ и $\mathfrak{d}_{ij\,km} = ((i, j), (k, m))$. На рис. 8–11 приведены соответственно программа $\pi_1 \otimes \pi_2'$, фрагмент шкалы $\mathcal{F}^{0,0} \otimes \mathcal{F}^{0,0}$, фрагмент графа $(\pi_1 \otimes \pi_2') \oplus (\mathcal{F}^{0,0} \otimes \mathcal{F}^{0,0})$ и фрагмент графа $\Gamma^{\mathcal{F}^{0,0}}_{\pi_1,\pi_2'}$.

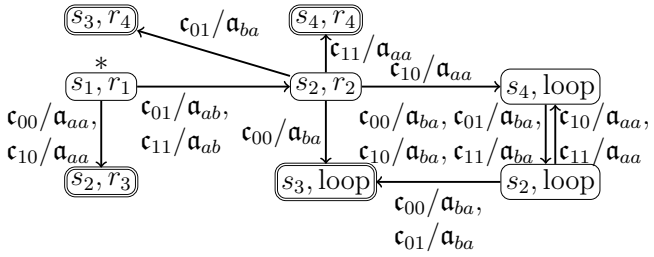

Рис. 8. Произведение программ (пример 11)

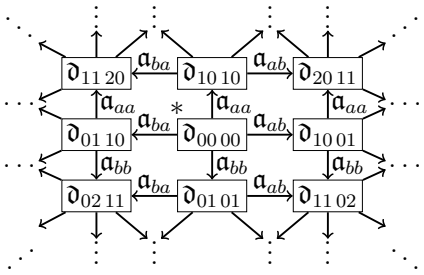

Рис. 9. Фрагмент произведения шкал (пример 11)

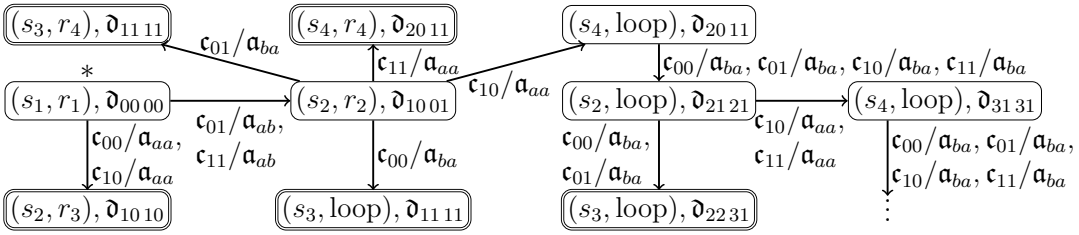

Рис. 10. Фрагмент графа вычислений произведения программ (пример 11)

Утверждение 11. Для любых совместных входных $\mathcal{F}$-трасс τ_1, τ_2 одинаковой длины программ π_1, π_2 соответственно путь $\tau_1 \otimes \tau_2$ — это входной путь в $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$.

Доказательство. Докажем утверждение для конечных трасс индукцией по длине, после чего — для бесконечных трасс.

Б а з а: τ_1 и τ_2 имеют длину 0. Тогда τ_i — это трасса из одной конфигурации (en_i, d^0), $i \in \{1, 2\}$, и $\tau_1 \otimes \tau_2$ — путь из одной вершины $((en_1, en_2), (d^0, d^0))$, являющейся входом графа $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$, то есть входной путь в $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$.

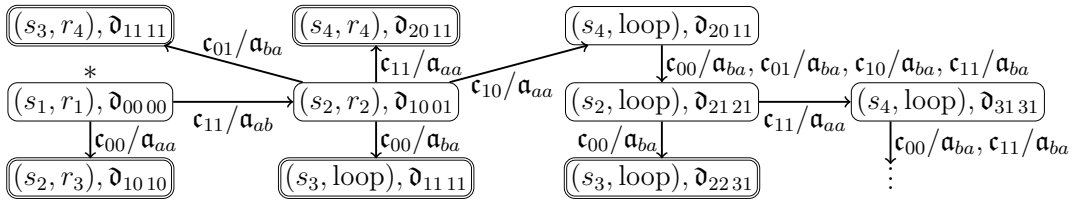

Рис. 11. Фрагмент графа совместных вычислений программ (пример 11)

Индуктивный переход: τ_1 и τ_2 имеют длину $n+1$, $n \in \mathbb{N}_0$, и $\rho' = \tau_1|^n \otimes \tau_2|^n$ — это входной путь в $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$. Положим для ясности, что $(s_i, d_i) \xrightarrow{c_i/a_i} (r_i, e_i)$ — последний переход трассы τ_i, $i \in \{1,2\}$. По устройству произведения трасс последней вершиной пути ρ' является $((s_1,s_2),(d_1,d_2))$. По устройству графа $(\pi_1 \otimes \pi_2) \oplus (\mathcal{F} \otimes \mathcal{F})$ в нём содержится дуга $t = (((s_1,s_2),(d_1,d_2)) \xrightarrow{(c_1,c_2)/(a_1,a_2)} ((r_1,r_2),(e_1,e_2)))$. Если $d_1 = d_2$, то по утверждению 8 $c_1 = c_2$. Значит, в любом случае дуга t содержится в $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$, тогда $\tau_1 \otimes \tau_2 = (\rho' \xrightarrow{(c_1,c_2)/(a_1,a_2)} ((r_1,r_2),(e_1,e_2)))$ — это входной путь в $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$.

Переход к бесконечным трассам: пусть τ_1 и τ_2 имеют бесконечную длину и утверждение справедливо для любых конечных трасс указанного в условии вида. Тогда для каждого $n \in \mathbb{N}_0$ в $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$ содержится входной путь $\rho_n = \tau_1|^n \otimes \tau_2|^n$ и ρ_{n+1} получается из ρ_n добавлением в конец одного перехода этого графа, обозначим этот переход t_n. Значит, в $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$ существует бесконечный путь ρ из входа по переходам $t_0, t_1, t_2, \ldots$ и по устройству произведения трасс $\rho = \tau_1 \otimes \tau_2$. ■

Утверждение 12. Любой входной путь ρ в $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$ представим в виде $\rho = \tau_1 \otimes \tau_2$, где τ_1 и τ_2 — совместные входные $\mathcal{F}$-трассы программ π_1, π_2 соответственно.

Доказательство. Докажем утверждение для конечных путей индукцией по длине, после чего — для бесконечных путей.

База: ρ имеет длину 0. Тогда $\rho = \tau_1 \otimes \tau_2$, где τ_i состоит из одной вершины (en_i, d^0), $i \in \{1,2\}$, и по утверждению 8 τ_1 и τ_2 — совместные входные $\mathcal{F}$-трассы программ π_1, π_2 соответственно.

Индуктивный переход: ρ имеет длину $n+1$, $n \in \mathbb{N}_0$, и существуют совместные входные $\mathcal{F}$-трассы τ_1', τ_2' программ π_1, π_2, удовлетворяющие равенству $\rho|^n = \tau_1' \otimes \tau_2'$. Положим, что $((s_1,s_2),(d_1,d_2)) \xrightarrow{(c_1,c_2)/(a_1,a_2)} ((r_1,r_2),(e_1,e_2))$ — последняя дуга пути ρ. Для каждого $i \in \{1,2\}$ верно следующее. По определению произведения трасс трасса τ_i' ведёт в конфигурацию (s_i, d_i). По устройству графа вычислений, программы $\pi_1 \otimes \pi_2$ и шкалы $\mathcal{F} \otimes \mathcal{F}$ переход $(s_i, d_i) \xrightarrow{c_i/a_i} (r_i, e_i)$ входит в граф $\pi_i \oplus \mathcal{F}$, а значит, $\tau_i = (\tau_i' \xrightarrow{c_i/a_i} (r_i, e_i))$ — входная $\mathcal{F}$-трасса программы π_i. По утверждению 6 и уравновешенности шкалы $\mathcal{F}$ состояние данных d_i отличается от состояний данных остальных конфигураций трасс τ_1' и τ_2', кроме, быть может, состояния d_{3-i}. По определению графа совместных вычислений если $d_1 = d_2$, то $c_1 = c_2$. По последнему соотношению, утверждению 8 и совместности трасс τ_1', τ_2' трассы $\tau_1 = (\tau_1' \xrightarrow{c_1/a_1} (r_1, e_1))$ и $\tau_2 = (\tau_2' \xrightarrow{c_2/e_2} (r_2, e_2))$ совместны. По определению произведения трасс $\rho = \tau_1 \otimes \tau_2$.

Переход к бесконечным путям: ρ имеет бесконечную длину; полагаем, что утверждение справедливо для любых конечных путей указанного в условии вида. Тогда для каждого $n \in \mathbb{N}_0$ путь $\rho_n = \rho|^n$ представим в виде $\rho_n = \tau_1^n \otimes \tau_2^n$, где τ_1^n и τ_2^n — совместные входные $\mathcal{F}$-трассы программ π_1, π_2, и для каждого $i \in \{1,2\}$

трасса τ_i^{n+1} получается из τ_i^n добавлением в конец одного перехода программы π_i (обозначим этот переход t_i^n), значит, бесконечный путь τ_i из конфигурации (en_i, d^0) по переходам $t_i^0, t_i^1, t_i^2, \ldots$ является бесконечной входной $\mathcal{F}$-трассой программы π_i. При этом трассы τ_1 и τ_2 совместны — иначе по утверждению 10 существуют несовместные конечные начальные отрезки этих трасс и по утверждению 9 для некоторого n несовместны трассы τ_1^n и τ_2^n, чего быть не может по полученному выше. ■

Вершину $((s_1, s_2), (d_1, d_2))$ графа $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$ назовём *достижимой* в этом графе, если она достижима из входа, и *опровергающей*, если верно одно из следующих условий:

1) $s_1 \in EX_1$ и $s_2 \notin EX_2$;
2) $s_1 \notin EX_1$ и $s_2 \in EX_2$;
3) $s_1 \in EX_1$, $s_2 \in EX_2$ и $d_1 \neq d_2$.

Пример 12. Среди вершин графа совместных вычислений на рис. 11 опровергающими являются все выходы, кроме $((s_3, r_4), \mathfrak{d}_{1111})$, и только они. В этом графе содержится бесконечно много неизображённых опровергающих вершин — например, недостижимая вершина $((s_3, r_4), \mathfrak{d}_{2002})$.

Утверждение 13. Для любой достижимой вершины (ss, dd) графа $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$ верно $dd \in \mathcal{B}_{\mathcal{F}}$.

Доказательство. Следует из утверждений 7 и 12. ■

Утверждение 14. Соотношение $\pi_1 \nsim_{\mathcal{F}} \pi_2$ верно тогда и только тогда, когда в $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$ существует достижимая опровергающая вершина.

Доказательство.

Необходимость. Пусть $\pi_1 \nsim_{\mathcal{F}} \pi_2$. По утверждению 3 существуют совместные $\mathcal{F}$-вычисления τ_1, τ_2 программ π_1, π_2 соответственно, имеющие различные результаты. Это означает, в частности, что одно из этих вычислений конечно. Положим без ограничения общности, что длина n трассы τ_1 конечна и не превосходит длину τ_2 (иначе достаточно поменять местами индексы 1 и 2). По утверждению 9 трассы τ_1 и $\tau_2' = \tau_2|^n$ совместны. По утверждению 11 $\rho = \tau_1 \otimes \tau_2'$ — входной путь в $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$, по устройству произведения трасс этот путь конечен. Положим, что ρ ведёт в вершину $v = ((s_1, s_2), (d_1, d_2))$. Значит, вершина v достижима в $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$. По выбору τ_1 верно $s_1 \in EX_1$. По устройству произведения трасс и неравенству результатов вычислений τ_1 и τ_2, если $s_2 \in EX_2$, то $d_1 \neq d_2$. Значит, в любом случае вершина v является опровергающей.

Достаточность. Положим, что в $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$ по некоторому пути ρ достижима некоторая опровергающая вершина $v = ((s_1, s_2), (d_1, d_2))$. По утверждению 12 и устройству произведения трасс существуют конечные совместные входные трассы τ_1, τ_2 одинаковой длины (обозначим её n) программ π_1, π_2, ведущие в конфигурации (s_1, d_1) и (s_2, d_2) соответственно.

Случай 1: $s_1 \in EX_1$ и $s_2 \in EX_2$. Тогда d_1 и d_2 — результаты совместных $\mathcal{F}$-вычислений τ_1, τ_2 соответственно. Так как вершина v является опровергающей, верно $d_1 \neq d_2$. Значит, по утверждению 3 $\pi_1 \nsim_{\mathcal{F}} \pi_2$.

Случай 2: $s_1 \in EX_1$ и $s_2 \notin EX_2$. По совместности трасс τ_1, τ_2 существует модель $\mathcal{I}$, в которой реализуются эти трассы. Положим, что τ_2' — $\mathcal{I}$-вычисление программы π_2. Так как $s_1 \in EX_1$, верно $\mathcal{I}(\pi_1) = d_1$. Так как $s_2 \notin EX_w$, длина τ_2' больше длины τ_2. Если длина τ_2' бесконечна, то $\mathcal{I}(\pi_2) = \bot \neq d_1 = \mathcal{I}(\pi_1)$. Иначе длина τ_2' конечна, по утверждению 6 $\mathcal{I}(\pi_1) = \mathcal{F}(h_1)$ и $\mathcal{I}(\pi_2) = \mathcal{F}(h_2)$, где h_1 и h_2 — цепочки трасс τ_1 и τ_2', и из этого, неравенства $|h_1| < |h_2|$ (так как τ_2' длиннее τ_2 и длина τ_2 равна длине τ_1) и уравновешенности шкалы $\mathcal{F}$ следует $\mathcal{I}(\pi_1) \neq \mathcal{I}(\pi_2)$, а значит, $\pi_1 \nsim_{\mathcal{F}} \pi_2$.

Случай 3: $s_1 \notin EX_1$ и $s_2 \in EX_2$ — повторяет случай 2 с взаимной заменой индексов 1 и 2. ■

4. Критериальная система

Для шкалы $\mathcal{F} = (D, d^0, \circ)$ и $(\mathfrak{A} \times \mathfrak{A})$-шкалы $\mathcal{W} = (W, w^0, \odot)$ *критериальным морфизмом* $\varphi : \mathcal{F} \to \mathcal{W}$ будем называть функцию $\varphi : \mathcal{B}_\mathcal{F} \to W$, заданную так:

— $\varphi(d^0, d^0) = w^0$;
— если $(d, e) \in \mathcal{B}_\mathcal{F}$ и $a, b \in \mathfrak{A}$, то $\varphi(d, e) \odot (a, b) = \varphi(d \circ a, e \circ b)$.

Записью $\mathcal{E}_\varphi$ будем обозначать множество $\{\varphi(d, d) : d \in \mathcal{R}_\mathcal{F}\}$. *Критериальной системой* шкалы $\mathcal{F} = (D, d^0, \circ)$ назовём пару $(\mathcal{W}, \varphi)$, где $\mathcal{W}$ — $(\mathfrak{A} \times \mathfrak{A})$-шкала, $\varphi : \mathcal{F} \to \mathcal{W}$ и для любой пары $(d, e) \in \mathcal{B}_\mathcal{F}$ из соотношения $\varphi(d, e) \in \mathcal{E}_\varphi$ следует $d = e$. Шкалу $\mathcal{W}$ системы $\mathcal{K}$ будем называть *критериальной* и её состояния — *критериями*; систему $\mathcal{K}$ — $\mathfrak{k}$-*ограниченной*, если для любых цепочек h_1, h_2 одинаковой длины существует не более $\mathfrak{k}$ критериев w, для которых $\mathcal{W}(w, h_1, h_2) \in \mathcal{E}_\varphi$.

Пример 13. Рассмотрим шкалу $\mathcal{F}^{0,0}$ с операцией $\circ$ из примера 2, $(\mathfrak{A} \times \mathfrak{A})$-шкалу $\mathcal{W} = (\mathbb{Z}, 0, \odot)$, где $m \odot (a, a) = m \odot (b, b) = m$, $m \odot (a, b) = m + 1$ и $m \odot (b, a) = m - 1$, и функцию $\varphi : \mathcal{B}_\mathcal{F} \to \mathbb{Z}$, заданную равенством $\varphi((n_1, m_1), (n_2, m_2)) = n_1 - n_2$. Тогда нетрудно убедиться, что $(\mathcal{W}, \varphi)$ — 1-ограниченная критериальная система для $\mathcal{F}^{0,0}$:

— $\varphi((0, 0), (0, 0)) = 0$;
— $\varphi((n_1, m_1), (n_2, m_2)) \odot (a, a) = n_1 - n_2 = \varphi((n_1, m_1) \circ a, (n_2, m_2) \circ a)$;
— $\varphi((n_1, m_1), (n_2, m_2)) \odot (b, b) = n_1 - n_2 = \varphi((n_1, m_1) \circ b, (n_2, m_2) \circ b)$;
— $\varphi((n_1, m_1), (n_2, m_2)) \odot (a, b) = (n_1 + 1) - n_2 = \varphi((n_1, m_1) \circ a, (n_2, m_2) \circ b)$;
— $\varphi((n_1, m_1), (n_2, m_2)) \odot (b, a) = n_1 - (n_2 + 1) = \varphi((n_1, m_1) \circ b, (n_2, m_2) \circ a)$;
— $\mathcal{E}_\varphi = \{0\}$ и при этом $\varphi((n, m), (n, m)) = n - n = 0$;
— для любых цепочек h_1 и h_2, содержащих соответственно n_1 и n_2 букв a, только для критерия $w = (n_2 - n_1)$ верно $\mathcal{W}(w, h_1, h_2) = 0$.

Утверждение 15. Для любой критериальной системы $(\mathcal{W}, \varphi)$ любой шкалы $\mathcal{F}$ и для любой пары $(d, e) \in \mathcal{B}_\mathcal{F}$ верно следующее: $d = e$ тогда и только тогда, когда $\varphi(d, e) \in \mathcal{E}_\varphi$.

Доказательство. Необходимость следует из определения множества $\mathcal{E}_\varphi$, достаточность — из определения критериальной системы. ■

При обсуждении алгоритмов, использующих критериальную систему $\mathcal{K} = (\mathcal{W}, \varphi)$, где $\mathcal{W} = (W, w^0, \odot)$, будем полагать заранее заданной *алгоритмическую составляющую* этой системы — представление элементов шкалы $\mathcal{W}$, алгоритмы $\mathcal{A}_0^\mathcal{K}$, $\mathcal{A}_e^\mathcal{K}$, $\mathcal{A}_\odot^\mathcal{K}$, $\mathcal{A}_=^\mathcal{K}$ и $\mathcal{A}_\leftarrow^\mathcal{K}$ и функции $\mathfrak{f}_e^\mathcal{K}, \mathfrak{f}_\odot^\mathcal{K}, \mathfrak{f}_=^\mathcal{K}, \mathfrak{f}_\leftarrow^\mathcal{K} : \mathbb{N}_0 \to \mathbb{N}_0$ следующего вида:

— $\mathcal{A}_0^\mathcal{K}() = w^0$;
— $\mathcal{A}_e^\mathcal{K}(w) =$ «Да», если $w \in \mathcal{E}_\varphi$, и «Нет» иначе; $\mathfrak{f}_e^\mathcal{K}(n)$ — сложность этого алгоритма на значениях вида $w = \mathcal{W}(h_1, h_2)$, где $|h_1| = |h_2| \leqslant n$;
— $\mathcal{A}_\odot^\mathcal{K}(w, a, b) = w \odot (a, b)$; $\mathfrak{f}_\odot^\mathcal{K}(n)$ — сложность этого алгоритма на значениях вида $w = \mathcal{W}(h_1, h_2)$, где $|h_1| = |h_2| \leqslant n$;
— $\mathcal{A}_=^\mathcal{K}(w, u) =$ «Да», если $w = u$, и «Нет» иначе; $\mathfrak{f}_=^\mathcal{K}(n)$ — сложность этого алгоритма на значениях вида $w = \mathcal{W}(h_1, h_2)$ и $u = \mathcal{W}(g_1, g_2)$, где $|h_1| = |h_2| \leqslant n$, $|g_1| = |g_2| \leqslant n$;
— выполнение $\mathcal{A}_\leftarrow^\mathcal{K}(x, w)$ устанавливает (копирует) в x значение w; $\mathfrak{f}_\leftarrow^\mathcal{K}(n)$ — сложность этого алгоритма на значениях вида $w = \mathcal{W}(h_1, h_2)$, где $|h_1| = |h_2| \leqslant n$.

Сложностной характеристикой критериальной системы для такой алгоритмической составляющей будем называть функцию $\mathfrak{f}^\mathcal{K} : \mathbb{N}_0 \to \mathbb{N}_0$, задающуюся равенством

$\mathfrak{f}^{\mathcal{K}}(n) = \max(\mathfrak{f}^{\mathcal{K}}_e(n), \mathfrak{f}^{\mathcal{K}}_{\odot}(n), \mathfrak{f}^{\mathcal{K}}_{=}(n), \mathfrak{f}^{\mathcal{K}}_{\leftarrow}(n))$. Далее для наглядности вместо вызовов вспомогательных алгоритмов будем записывать соответственно значения и соотношения w^0, $w \in \mathcal{E}_\varphi$, $w \odot (a, b)$, $w = u$ и описание установки критерия.

Пример 14. Для критериальной системы из примера 13 с алгоритмической составляющей, естественно отвечающей определению этой критериальной системы, сложностная характеристика имеет порядок $O(1)$, так как такой порядок сложности имеют все требуемые алгоритмы: проверка равенства числа-критерия нулю, прибавление или вычитание единицы в зависимости от пары операторов, проверка равенства критериев, копирование числа.

5. Критериальный граф

Предположим заданными полные программы $\pi_1 = (S_1, en_1, EX_1, T_1)$ и $\pi_2 = (S_2, en_2, EX_2, T_2)$, уравновешенную шкалу $\mathcal{F} = (D, d^0, \circ)$ и её критериальную систему $\mathcal{K} = (\mathcal{W}, \varphi)$, где $\mathcal{W} = (W, w^0, \odot)$. Все утверждения формулируются для произвольных таких π_1, π_2, $\mathcal{F}$ и $\mathcal{K}$.

Критериальным графом программ π_1, π_2 и системы $\mathcal{K}$ назовём подграф $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$ графа $(\pi_1 \otimes \pi_2) \oplus \mathcal{W}$, содержащий все вершины и все дуги, кроме дуг $((s_1, s_2), w) \xrightarrow{(c_1,c_2)/(a_1,a_2)} v$, в которых $w \in \mathcal{E}_\varphi$ и $c_1 \neq c_2$. Вершины критериального графа будем называть *узлами*, пути в нём — *маршрутами*, значения s_1, s_2 и w — соответственно *состояниями* и *критерием узла* $((s_1, s_2), w)$. Назовём *φ-образом* вершины $v = ((s_1, s_2), (d_1, d_2))$ графа $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$ и пути $v_0 \xrightarrow{(c^1_1,c^1_2)/(a^1_1,a^1_2)} v_1 \xrightarrow{(c^2_1,c^2_2)/(a^2_1,a^2_2)} \dots$ в этом графе соответственно набор $\varphi(v) = ((s_1, s_2), \varphi(d_1, d_2))$ и путь $\varphi(v_0) \xrightarrow{(c^1_1,c^1_2)/(a^1_1,a^1_2)} \varphi(v_1) \xrightarrow{(c^2_1,c^2_2)/(a^2_1,a^2_2)} \dots$

Пример 15. Рассмотрим программы π_1, π'_2 и критериальную систему $\mathcal{K} = (\mathcal{W}, \varphi)$ из примеров 11 и 13. На рис. 12 показан фрагмент графа $\Gamma^{\mathcal{K}}_{\pi_1,\pi'_2}$, содержащий φ-образы всех вершин на рис. 11.

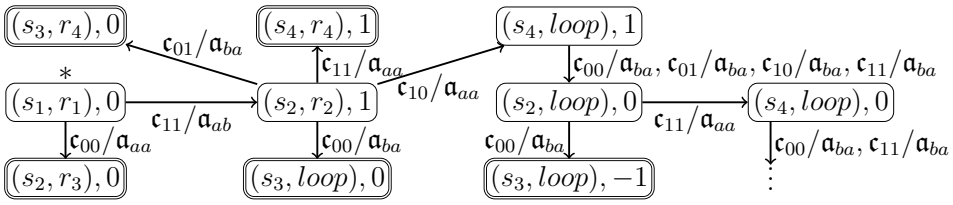

Рис. 12. Фрагмент критериального графа (пример 15)

Утверждение 16. Входными путями в $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$ являются все φ-образы входных путей в $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$, и только они.

Доказательство. Достаточно обосновать следующее: 1) φ-образы всех достижимых вершин графа $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$ являются вершинами графа $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$; 2) φ-образом входа в $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$ является вход в $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$; 3) метки дуг, исходящих в $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$ из достижимой вершины v, совпадают с метками дуг, исходящих из $\varphi(v)$ в $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$; 4) если в $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$ дуга из достижимой вершины v заходит в δ, то в $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$ дуга из $\varphi(v)$ с той же меткой заходит в $\varphi(\delta)$. Всё это следует из определений графов $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$, $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$ и критериальной системы и утверждений 13 и 15. ■

Узел $((s_1, s_2), w)$ графа $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$ назовём *достижимым* в этом графе, если он достижим из входа, *нейтральным*, если $w \in \mathcal{E}_\varphi$, и *опровергающим*, если верно одно из следующих условий:

1) $s_1 \in EX_1$ и $s_2 \notin EX_2$;
2) $s_1 \notin EX_1$ и $s_2 \in EX_2$;
3) $s_1 \in EX_1$, $s_2 \in EX_2$ и $w \notin \mathcal{E}_\varphi$.

Пример 16. С учётом изложенного в примере 13 среди узлов критериального графа на рис. 12 нейтральными являются все помеченные числом 0, и только они, опровергающими — все выходы, кроме $((s_3, r_4), 0)$. В этом графе содержится бесконечно много неизображённых опровергающих узлов — например, недостижимый узел $((s_3, r_4), 2)$.

Утверждение 17. Достижимыми опровергающими узлами в $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$ являются φ-образы всех достижимых опровергающих вершин графа $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$, и только они.

Доказательство. С учётом утверждения 16 и устройства φ-образов путей достаточно показать, что достижимая вершина v графа $\Gamma^{\mathcal{F}}_{\pi_1,\pi_2}$ является опровергающей тогда и только тогда, когда узел $\varphi(v)$ опровергающий. Положим, что $v = ((s_1, s_2), (d_1, d_2))$. Тогда по утверждению 13 верно $(d_1, d_2) \in \mathcal{B}_{\mathcal{F}}$, а значит, $\varphi(v) = ((s_1, s_2), w)$, где $w = \varphi(d_1, d_2)$. Осталось заметить, что состояния программ в v и $\varphi(v)$ одинаковы и из утверждения 15 следует равносильность $(d_1 = d_2 \Leftrightarrow w \in \mathcal{E}_\varphi)$. ■

Утверждение 18. Соотношение $\pi_1 \nsim_{\mathcal{F}} \pi_2$ верно тогда и только тогда, когда в $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$ существует достижимый опровергающий узел.

Доказательство. Следует из утверждений 17 и 14. ■

Утверждение 19. Если система $\mathcal{K}$ $\mathfrak{k}$-ограниченная, в $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$ достижимы узлы $v_1, \ldots, v_{\mathfrak{k}+1}$, где $v_i = ((s_1, s_2), w_i)$, $i \in \{1, \ldots, \mathfrak{k}+1\}$, хотя бы одно из состояний s_1, s_2 завершаемо и критерии w_i, $i \in \{1, \ldots, \mathfrak{k}+1\}$, попарно различны, то $\pi_1 \nsim_{\mathcal{F}} \pi_2$.

Доказательство. Положим, что $\|s_1\|_{\pi_1} \leqslant \|s_2\|_{\pi_2}$ (если $\|s_1\|_{\pi_1} > \|s_2\|_{\pi_2}$, то далее достаточно поменять местами индексы 1 и 2). Тогда в π_1 существует путь из s_1 в выход. Рассмотрим кратчайший такой путь $p_0 \xrightarrow{c_1/a_1} p_1 \xrightarrow{c_2/a_2} \ldots \xrightarrow{c_n/a_n} p_n$, $p_0 = s_1$, $p_n \in EX_1$. Тогда по устройству критериального графа, утверждению 12 и неравенству в начале доказательства в $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$ существуют пути $v_i \xrightarrow{(c_1,c_1)/(a_1,b_1)} ((p_1, q_1), w_1^i) \xrightarrow{(c_2,c_2)/(a_2,b_2)} \ldots \xrightarrow{(c_n,c_n)/(a_n,b_n)} ((p_n, q_n), w_n^i)$, $i \in \{1, \ldots, \mathfrak{k}+1\}$, для некоторых $q_1, \ldots, q_n, b_1, \ldots, b_n, w_1^1, \ldots, w_n^1, \ldots, w_1^{\mathfrak{k}+1}, \ldots, w_n^{\mathfrak{k}+1}$ и при этом $w_n^i = \mathcal{W}(w_i, a_1 \ldots a_n, b_1 \ldots b_n)$. Значит, по $\mathfrak{k}$-ограниченности системы $\mathcal{K}$ хотя бы для одного $i \in \{1, \ldots, \mathfrak{k}+1\}$ верно $w_n^i \notin \mathcal{E}_\varphi$. Тогда хотя бы одна из вершин $v_i' = ((p_n, q_n), w_n^i)$ является опровергающей, при этом все вершины v_i' достижимы в $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$ и из утверждения 18 следует $\pi_1 \nsim_{\mathcal{F}} \pi_2$. ■

6. Алгоритм проверки эквивалентности программ

В описании и анализе алгоритма проверки эквивалентности программ (алгоритма 4) и вспомогательных алгоритмов 1–3 считаем заданными конечные множества операторов $\mathfrak{A}$ и логических условий $\mathfrak{C}$, шкалу $\mathcal{F} = (D, d^0, \circ)$ и её $\mathfrak{k}$-ограниченную критериальную систему $\mathcal{K} = (\mathcal{W}, \varphi)$ со шкалой $\mathcal{W} = (W, w^0, \odot)$ и некоторой алгоритмической составляющей. Для сохранения понятности изложения действия алгоритмов описываются «высокоуровнево» с использованием математической терминологии. В комментариях приводятся «низкоуровневые» детали, необходимые, в числе прочего, для оценки сложности алгоритмов.

Будем использовать следующие способы представления данных: $\mathfrak{A} = \{1, 2, \ldots, n_{\mathfrak{a}}\}$, $\mathfrak{C} = \{1, 2, \ldots, n_{\mathfrak{c}}\}$; множество состояний S каждой программы имеет вид $\{1, 2, \ldots, n_S\}$;

множество вида $\{1, 2, \ldots, n\}$ представляется числом n; одноместная функция $f : \{1, 2, \ldots, n\} \to X$ — вектором $(f(1), f(2), \ldots, f(n))$; множество $X \subseteq \{1, 2, \ldots, n\}$ — так же, как функция $f : \{1, 2, \ldots, n\} \to \{0, 1\}$, задающаяся равносильностью $f(x) = 1 \Leftrightarrow x \in X$. Двухместная функция $f : \{1, 2, \ldots, n_1\} \times \{1, 2, \ldots, n_2\} \to X$ (в том числе функция переходов программы) представляется матрицей, где $f(i, j)$ — значение элемента матрицы в i-й строке и j-м столбце. Последовательность нефиксированной длины задаётся списком элементов, за исключением цепочки: она представляется вектором элементов. Для цепочки переменной длины сразу выделяется столько ячеек памяти, сколько необходимо для хранения любой цепочки длины, равной наибольшему из размеров рассматриваемых программ. Конечное множество других видов по умолчанию представляется списком своих элементов — добавление элемента происходит в конец списка, проверка принадлежности элемента множеству состоит в проходе по списку с проверкой равенства.

Алгоритм 1. Пополнение программы

Вход: конечная программа $\pi = (S, en, EX, T)$.
Выход: пополнение π' программы π.

1: Произвольно выбрать значения $\mathrm{loop} \notin S$ и $a \in \mathfrak{A}$.
// По выбору представления множества состояний $\mathrm{loop} = n_S + 1$. *Для определённости выбирается* $a = 1$.
2: **Вернуть** $(S \cup \{\mathrm{loop}\}, en, EX, T^{\mathrm{loop},a})$.
// Матрицу $T^{\mathrm{loop},a}$ *можно вычислить так: установить во всех строках этой матрицы, кроме последней, те же значения, что и в* T, *а в каждой ячейке последней строки — значение* (loop, a). *Затем заменить каждое значение* $T^{\mathrm{loop},a}(s, c) = \bot$, *где* $s \notin EX$, *на* $T^{\mathrm{loop},a}(s, c) = (\mathrm{loop}, a)$.

Алгоритм 2. Вычисление завершаемых состояний программы

Вход: конечная полная программа $\pi = (S, en, EX, T)$.
Выход: множество S^f всех завершаемых состояний программы π.

1: Вычислить орграф G, обратный к орграфу π. Добавить в G произвольный простой цикл, содержащий все вершины множества EX и только их.
// Для определённости выбирается цикл, в котором выходы соединяются дугами по возрастанию и самый большой выход соединяется с самым маленьким. Граф G *представляется списком смежности. Вычисление графа* G *производится полным перебором ячеек матрицы* T *с добавлением соответствующих дуг и затем добавлением дуг цикла.*
2: Произвольно выбрать вершину $s \in EX$ и применить к ней и графу G поиск в ширину [27], вычисляющий множество X состояний, достижимых из s.
3: **Вернуть** $S^f = X$.

Алгоритм 3. Обход критериального графа

Вход: конечные полные программы $\pi_1 = (S_1, en_1, EX_1, T_1)$ и $\pi_2 = (S_2, en_2, EX_2, T_2)$ и соответствующие множества S_1^f и S_2^f всех завершаемых состояний этих программ.

Выход: ответ «Да», если $\pi_1 \sim_{\mathcal{F}} \pi_2$, и «Нет» иначе.

// В алгоритме используются следующие вспомогательные значения:

— *набор* $(s_1, s_2, w) \in S_1 \times S_2 \times W$, *в начале выполнения равен* (en_1, en_2, w^0);

— *разметка* $F : S_1 \times S_2 \to 2^W$, *в начале выполнения все значения* F *равны* $\varnothing$ *(используются только конечные подмножества* W*);*

— *множество* $X \subseteq \mathfrak{C} \times \mathfrak{C}$, *значение в начале выполнения неважно;*

— *конечная последовательность* P *элементов множества* $S_1 \times S_2 \times W \times 2^{\mathfrak{C}\times\mathfrak{C}}$, *в начале выполнения эта последовательность пуста.*

1: **Если**
а) ($s_1 \notin S_1^f$ и $s_2 \notin S_2^f$) или
б) $w \in F(s_1, s_2)$, **то**
2: **выход**, ответ «Да».
3: **Если**
а) ($s_1 \in EX_1$ и $s_2 \notin EX_2$) или ($s_1 \notin EX_1$ и $s_2 \in EX_2$) или ($s_1 \in EX_1$, $s_2 \in EX_2$ и $w \notin \mathcal{E}_\varphi$) или
б) $|F(s_1, s_2)| = \mathfrak{k}$, **то**
4: **выход**, ответ «Нет».
5: Добавить в множество $F(s_1, s_2)$ критерий w.
6: Вычислить множество пар $X \subseteq \mathfrak{C} \times \mathfrak{C}$: **если** $w \in \mathcal{E}_\varphi$, **то** $X = \{(c, c) : c \in \mathfrak{C}\}$, **иначе** $X = \mathfrak{C} \times \mathfrak{C}$.
// Для проверки $w \in \mathcal{E}_\varphi$ *используется результат проверки* $w \notin \mathcal{E}_\varphi$ *с шага 3.*
7: **Для всех** $(c_1, c_2) \in X$:
// Значения (c_1, c_2) *перебираются в порядке расположения в списке и удаляются из списка при рассмотрении.*
8: Вычислить значения $(a_1, r_1) = T_1(s_1, c_1)$ и $(a_2, r_2) = T_2(s_2, c_2)$.
9: Заменить набор (s_1, s_2, w) на $(r_1, r_2, w \odot (a_1, a_2))$.
// Перед заменой набор (s_1, s_2, w, X) *для текущего значения* X *добавляется в конец последовательности* P.
10: Выполнить тело алгоритма 3 (основной рекурсивный вызов).
Если его результат «Нет», **то выход**, ответ «Нет».
11: Восстановить значения (s_1, s_2, w) и X, которые были в начале выполнения шага 9.
// Значения берутся из последнего элемента P, *этот элемент удаляется из* P.
12: **Вернуть** «Да».

Лемма 1. Алгоритм 1 имеет сложность $O(\mathfrak{n})$, где $\mathfrak{n}$ — размер программы на входе.

Доказательство. На шаге 1 выполняется $O(1)$ действий. На шаге 2 перебираются $(\mathfrak{n}+1)n_{\mathfrak{c}} = O(\mathfrak{n})$ ячеек матрицы $T^{\mathrm{loop},a}$ и для каждой ячейки выполняется $O(1)$ действий. Значит, суммарно получаем $O(\mathfrak{n})$ действий. ■

Лемма 2. Для любой конечной программы π результат выполнения алгоритма 1 на входе π — это пополнение программы π.

Доказательство. Пусть $\pi = (S, en, EX, T)$. Алгоритм 1 выдаёт программу $(S \cup \{\text{loop}\}, en, EX, T^{\text{loop},a})$ для некоторых $\text{loop} \notin S$ и $a \in \mathfrak{A}$, то есть, по определению, пополнение $\pi^{\text{loop},a}$ программы π. ■

Лемма 3. Алгоритм 2 имеет сложность $O(\mathfrak{n})$, где $\mathfrak{n}$ — размер программы на входе.

Доказательство. На шаге 1 просматривается $\mathfrak{n}n_{\mathfrak{c}} = O(\mathfrak{n})$ ячеек матрицы T и для каждой ячейки выполняется $O(1)$ действий, после чего за $O(\mathfrak{n})$ действий добавляется цикл. Поиск в ширину на шаге 2 имеет сложность $O(\mathfrak{n})$, согласно [27] и тому, что программа имеет $O(\mathfrak{n})$ переходов. Значит, суммарно получаем $O(\mathfrak{n})$ действий. ■

Лемма 4. Для любой конечной полной программы π результат выполнения алгоритма 2 на входе π — это множество всех завершаемых состояний программы π.

Доказательство. Достижимость вершины t из $s \in EX$ на шаге 2 равносильна достижимости t хотя бы из одного выхода в графе, обратном к π, а это равносильно достижимости хотя бы одного выхода из t в π, то есть завершаемости t. ■

Пусть задано выполнение $\mathfrak{E}$ алгоритма 3. Будем называть *итерацией* выполнения $\mathfrak{E}$ однократное последовательное выполнение шагов 1–12, кроме действий в основных рекурсивных вызовах на шаге 10, записью $\mathfrak{E}(i)$ обозначим i-ю итерацию выполнения $\mathfrak{E}$ при нумерации с единицы с упорядочиванием по времени начала. *Узлом итерации* $\mathfrak{I}$ назовём значение $[\mathfrak{I}] = ((s_1, s_2), w)$ в начале итерации $\mathfrak{I}$. Итерацию $\mathfrak{I}_c$ будем считать *ребёнком* итерации $\mathfrak{I}_p$, если итерация $\mathfrak{I}_c$ начинается основным рекурсивным вызовом на шаге 10 итерации $\mathfrak{I}_p$, набор значений $(c_1, c_2, a_1, a_2, r_1, r_2)$ в начале этого шага будем называть *переходным набором* итерации $\mathfrak{I}_c$. *Итерационным путём* назовём путь вида $\mathfrak{I}_0 \xrightarrow{(c_1,\ell_1)/(a_1,b_1)} \mathfrak{I}_1 \xrightarrow{(c_2,\ell_2)/(a_2,b_2)} \dots$, где $\mathfrak{I}_{i+1}$ — ребёнок итерации $\mathfrak{I}_i$ и c_i, ℓ_i, a_i, b_i — первые четыре элемента переходного набора $\mathfrak{I}_{i+1}$. Итерационный путь будем называть *входным*, если он исходит из итерации $\mathfrak{E}(1)$. *Трасса итерационного пути* P — путь, получающийся из P заменой каждой итерации на её узел; *трасса итерации* $\mathfrak{I}$ — трасса входного итерационного пути, ведущего в $\mathfrak{I}$.

Лемма 5. Для любой итерации $\mathfrak{I}$ любого выполнения алгоритма 3 верно следующее: если $[\mathfrak{I}] = ((s_1, s_2), w)$ и в начале итерации $\mathfrak{I}$ имеет место $u \in F(s_1, s_2)$, то существует итерация $\mathfrak{I}'$ с меньшим номером, для которой $[\mathfrak{I}'] = ((s_1, s_2), u)$.

Доказательство. Получить элемент u в множестве $F(s_1, s_2)$ в начале итерации $\mathfrak{I}$ можно, только добавив его выполнением шага 5 до начала этой итерации, то есть на шаге 5 некоторой итерации $\mathfrak{I}'$ с меньшим номером. Для такого добавления u необходимо равенство $[\mathfrak{I}'] = ((s_1, s_2), u)$. ■

Лемма 6. В начале любой итерации любого выполнения алгоритма 3 верно $|F(s_1, s_2)| \leqslant \mathfrak{k}$, где s_1 и s_2 — состояния узла итерации.

Доказательство. В начале выполнения алгоритма $|F(s_1, s_2)| = 0$ для всех $s_1 \in S_1$ и $s_2 \in S_2$. Значения F изменяются только на шаге 5. При выполнении этого шага размер значения $F(s_1, s_2)$ для состояний s_1, s_2 узла итерации увеличивается на 1, а остальные значения F не изменяются. При этом увеличение размера $F(s_1, s_2)$ на шаге 5 возможно только в том случае, если на этой итерации не выполнено условие 3б, это возможно, только если $|F(s_1, s_2)| < \mathfrak{k}$, тогда размер этого множества после добавления элемента не превосходит $\mathfrak{k}$. ■

Лемма 7. Трасса любой итерации $\mathfrak{I}$ любого выполнения алгоритма 3 является входным маршрутом в $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$.

Доказательство. Рассмотрим произвольное выполнение $\mathfrak{E}$ алгоритма 3 и применим индукцию по структуре входного итерационного пути, ведущего в $\mathfrak{I}$.

Б а з а: если $\mathfrak{I} = \mathfrak{E}(1)$, то $[\mathfrak{I}] = ((en_1, en_2), w^0)$, трасса итерации $\mathfrak{I}$ состоит из одного узла, и этот узел — вход графа $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$.

И н д у к т и в н ы й п е р е х о д: $\mathfrak{I}$ — ребёнок итерации $\mathfrak{I}_p$, и трасса итерации $\mathfrak{I}_p$ является входным маршрутом в $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$. Пусть $[\mathfrak{I}_p] = ((s_1, s_2), w)$ и $(c_1, c_2, a_1, a_2, r_1, r_2)$ — переходный набор итерации $\mathfrak{I}$. По устройству шага 10 и соответствующим определениям трасса итерации $\mathfrak{I}$ получается из трассы итерации $\mathfrak{I}_p$ продолжением на одну дугу $t = (((s_1, s_2), w) \xrightarrow{(c_1,c_2)/(a_1,a_2)} ((r_1, r_2), w \odot (a_1, a_2)))$. Значит, достаточно показать, что t входит в граф $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$.

По устройству шага 9 $(a_1, r_1) = T_1(s_1, c_1)$ и $(a_2, r_2) = T_2(s_2, c_2)$. Значит, t входит в граф $(\pi_1 \otimes \pi_2) \oplus \mathcal{W}$ по определению этого графа. По устройству множества X на шаге 6 если $w \in \mathcal{E}_\varphi$, то $c_1 = c_2$. Значит, в любом случае t содержится в графе $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$ по определению этого графа. ■

В леммах 8–10 символом $\mathfrak{n}$ обозначено значение $\max(|\pi_1|, |\pi_2|)$ для программ π_1 и π_2, подающихся на вход алгоритму 3.

Лемма 8. В любом выполнении $\mathfrak{E}$ алгоритма 3 содержится $O(\mathfrak{n}^2)$ итераций.

Доказательство. Если на итерации $\mathfrak{I}$ выполняется шаг 5, то все значения F, кроме $F(s_1, s_2)$, где s_1 и s_2 — состояния узла $[\mathfrak{I}]$, не изменяются, а размер множества $F(s_1, s_2)$ увеличивается на 1 (так как если выполняется шаг 5, то не верно условие 1б, т. е. добавляемый критерий не содержится в $F(s_1, s_2)$). Из этого и леммы 6 следует, что в $\mathfrak{E}$ шаг 5 выполняется не более $\mathfrak{k}\,\mathfrak{n}^2$ раз. На каждой итерации шаги 5 и 7 либо оба не выполняются, либо оба выполняются по одному разу. На шаге 7 не более $|X|$ раз, т. е. не более $|\mathfrak{C}|^2$ раз, выполняется основной рекурсивный вызов, отвечающий одному очередному ребёнку. Следовательно, не более $\mathfrak{k}\,\mathfrak{n}^2$ итераций имеют детей и каждая такая итерация имеет не более $|\mathfrak{C}|^2$ детей. Значит, всего в $\mathfrak{E}$ содержится не более $(1 + |\mathfrak{C}|^2\,\mathfrak{k}\,\mathfrak{n}^2) = O(\mathfrak{n}^2)$ итераций. ■

Лемма 9. Критерий w узла любой итерации $\mathfrak{I}$ любого выполнения $\mathfrak{E}$ алгоритма 3 представим в виде $w = \mathcal{W}(h_1, h_2)$, где $h_1, h_2 \in \mathfrak{A}^*$ и $|h_1| = |h_2| = O(\mathfrak{n}^2)$.

Доказательство. Если $\mathfrak{I} = \mathfrak{E}(1)$, то $[\mathfrak{I}] = ((en_1, en_2), w^0)$, $w^0 = \mathcal{W}(\lambda, \lambda)$ и длины цепочек λ равны 0. Если $\mathfrak{I}$ — ребёнок итерации $\mathfrak{I}_p$, $[\mathfrak{I}_p] = ((s_1, s_2), \mathcal{W}(g_1, g_2))$ и $(a_1, a_2, c_1, c_2, r_1, r_2)$ — переходный набор итерации $\mathfrak{I}$, то $[\mathfrak{I}] = ((r_1, r_2), \mathcal{W}(g_1 a_1, g_2 a_2))$, и если длины цепочек g_1 и g_2 равны k, то длины цепочек $g_1 a_1$ и $g_2 a_2$ равны $(k + 1)$. Значит, критерий w узла любой итерации $\mathfrak{I}$ представим в виде $w = \mathcal{W}(h_1, h_2)$, где длины цепочек h_1, h_2 равны длине входного итерационного пути, ведущего в $\mathcal{I}$. Длина этого пути оценивается как $O(\mathfrak{n}^2)$ по лемме 8 и попарной различности итераций в любом итерационном пути. ■

Лемма 10. Алгоритм 3 имеет сложность $O(\mathfrak{n}^2 f(\mathfrak{n}^2))$ для некоторой функции $f : \mathbb{N}_0 \to \mathbb{N}_0$, удовлетворяющей равенству $f(m) = \mathfrak{f}^{\mathcal{K}}(O(m))$, где $\mathfrak{f}^{\mathcal{K}}$ — сложностная характеристика алгоритмической составляющей системы $\mathcal{K}$, используемой в алгоритме.

Доказательство. По лемме 8 в любом выполнении $\mathfrak{E}$ алгоритма 3 содержится $O(\mathfrak{n}^2)$ итераций. Следовательно, достаточно показать, что любая итерация $\mathfrak{I}$ выполнения $\mathfrak{E}$ имеет сложность $O(f(\mathfrak{n}^2))$ для подходящей функции f.

Положим, что $[\mathfrak{I}] = ((s_1, s_2), w)$. По лемме 9 $w = \mathcal{W}(h_1, h_2)$ для некоторых $h_1, h_2 \in \mathfrak{A}^*$, удовлетворяющих равенствам $|h_1| = |h_2| = O(\mathfrak{n}^2)$, и из этого и устройства алгоритма 3 и его шага 5 следует, что все элементы множества $F(s_1, s_2)$ представимы в виде $\mathcal{W}(g_1, g_2)$, где $g_1, g_2 \in \mathfrak{A}^*$ и $|g_1| = |g_2| = O(\mathfrak{n}^2)$. Положим $f(m) = \mathfrak{f}^{\mathcal{K}}(\widetilde{f}(m))$, где $\widetilde{f}(\mathfrak{n}^2) = O(\mathfrak{n}^2)$ — обозначенная выше оценка длин цепочек h_1, h_2, g_1, g_2. По лемме 6 на каждой итерации проверка условия 3б содержит не более $\mathfrak{k} = O(1)$ проверок равенства критериев и при этой проверке вычисляемый размер множества не превосходит $\mathfrak{k} = O(1)$. На этом основываются оценки сложности, изложенные далее.

Условие 1а проверяется за $O(1)$ действий, 1б — за $O(f(\mathfrak{n}^2))$, 3а — за $O(f(\mathfrak{n}^2))$, 3б — за $O(1)$ действий. Шаг 5 выполняется за $O(f(\mathfrak{n}^2))$ действий. На шаге 6 за $O(1)$ действий вычисляется множество X, содержащее $O(1)$ элементов, и для каждого элемента на шагах 7–11 итерации выполняется $O(f(\mathfrak{n}^2))$ действий. Значит, суммарно на всех шагах итерации выполняется $O(f(\mathfrak{n}^2))$ действий. ■

Лемма 11. Для любых конечных полных программ π_1 и π_2 и множеств их завершаемых состояний S_1^f и S_2^f выполнение алгоритма 3 на входе $(\pi_1, \pi_2, S_1^f, S_2^f)$ имеет результат «Нет» тогда и только тогда, когда $\pi_1 \nsim_{\mathcal{F}} \pi_2$.

Доказательство. Обозначим символом $\mathfrak{E}$ выполнение алгоритма 3 на входе $(\pi_1, \pi_2, S_1^f, S_2^f)$.

Необходимость. Положим, что выполнение $\mathfrak{E}$ имеет результат «Нет». Пусть $\mathfrak{E}(n)$ — последняя итерация выполнения $\mathfrak{E}$ и $[\mathfrak{E}(n)] = ((s_1, s_2), w)$. Из ответа «Нет» и устройства алгоритма следует, что на итерации $\mathfrak{E}(n)$ не выполнены условия шага 1 и выполнено хотя бы одно из условий шага 3. По лемме 7 в $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$ достижим узел $[\mathfrak{E}(n)]$.

Если на итерации $\mathfrak{E}(n)$ выполнено условие 3а, то узел $[\mathfrak{E}(n)]$ опровергающий и соотношение $\pi_1 \nsim_{\mathcal{F}} \pi_2$ следует из утверждения 18. Далее полагаем, что на $\mathfrak{E}(n)$ выполнено $|F(s_1, s_2)| = \mathfrak{k}$. Так как не выполнено условие 1б, все элементы $F(s_1, s_2)$ отличны от w (и попарно различны). По лемме 5 существуют итерации выполнения $\mathfrak{E}$ с узлами $((s_1, s_2), u)$ для всех $u \in F(s_1, s_2)$. По лемме 7 все эти узлы достижимы в $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$. Так как не выполнено условие 1а, хотя бы одно из состояний s_1, s_2 завершаемо. Тогда соотношение $\pi_1 \nsim_{\mathcal{F}} \pi_2$ следует из утверждения 19.

Достаточность. Положим, что $\pi_1 \nsim_{\mathcal{F}} \pi_2$. По устройству алгоритма 3 достаточно показать: а) на шаге 2 итерации $\mathfrak{E}(1)$ не выдаётся ответ «Да»; б) на шаге 4 хотя бы одной итерации $\mathfrak{I}$ выполнения $\mathfrak{E}$ выдаётся ответ «Нет» — тогда, согласно шагу 10, на всех итерациях входного итерационного пути в $\mathfrak{I}$ выдаётся ответ «Нет», в том числе на итерации $\mathfrak{E}(1)$, начинающей этот путь и предоставляющей ответ алгоритма. Покажем это:

а) Условие 1а не выполнено на итерации $\mathfrak{E}(1)$ — иначе входы обеих программ незавершаемы и все вычисления этих программ имеют результат $\perp$, а значит, программы $\mathcal{F}$-эквивалентны, что противоречит предположению достаточности. По описанию алгоритма в начале итерации $\mathfrak{E}(1)$ верно $F(s_1, s_2) = \varnothing$. Значит, и условие 1б не выполнено на итерации $\mathfrak{E}(1)$, и ответ «Да» на шаге 1 этой итерации не выдаётся.

б) По утверждению 18 в графе $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$ существует входной маршрут $\rho = (v_0 \xrightarrow{(c_1,\ell_1)/(a_1,b_1)} v_1 \xrightarrow{(c_2,\ell_2)/(a_2,b_2)} \ldots \xrightarrow{(c_n,\ell_n)/(a_n,b_n)} v_n)$ в некоторый опровергающий узел v_n. Положим, что $v_i = ((s_i, r_i), w_i)$ и $s_n \in EX_1$ (иначе по определению опровергающего узла $r_n \in EX_2$ и в дальнейших рассуждениях достаточно поменять ролями (s_i, c_i, a_i) и (r_i, ℓ_i, b_i) и индексы 1 и 2, относящиеся к программам). Возможны два случая:

С л у ч а й 1: v_n является узлом какой-либо итерации выполнения $\mathfrak{E}$. Рассмотрим итерацию $\mathfrak{E}(m)$ с наименьшим номером m, узлом которой является v_n. Условие 1а не выполнено на $\mathfrak{E}(m)$, так как s_n завершаемо. Условие 1б также не выполнено на $\mathfrak{E}(m)$ — иначе по лемме 5 существовала бы итерация $\mathfrak{E}(m')$ с номером $m' < m$ и узлом v_n, чего не может быть по выбору m. Значит, на итерации $\mathfrak{E}(m)$ на шаге 2 не выдаётся ответ «Да» и выполняется шаг 3. Так как узел v_n опровергающий, на итерации $\mathfrak{E}(m)$ выполнено условие 3а и на шаге 4 выдаётся ответ «Нет».

С л у ч а й 2: v_n не является узлом ни одной итерации выполнения $\mathfrak{E}$. Рассмотрим наименьший номер k, для которого v_{k+1} не является узлом ни одной итерации выполнения $\mathfrak{E}$. Так как $v_0 = [\mathfrak{E}(1)]$, то $k \in \{0, 1, \ldots, n-1\}$. Рассмотрим итерацию $\mathfrak{E}(m)$ с наименьшим номером m, узлом которой является v_k. По утверждениям 16, 12 и 1 в π_1 существует путь из s_k в s_n. Значит, состояние s_k завершаемо и условие 1а не выполнено на $\mathfrak{E}(m)$. Условие 1б не выполнено на $\mathfrak{E}(m)$ по тем же соображениям, что и в случае 1. Значит, на шаге 2 итерации $\mathfrak{E}(m)$ не выдаётся ответ «Да».

Предположим от противного, что на шаге 4 итерации $\mathfrak{E}(m)$ не выдаётся ответ «Нет». Тогда на итерации $\mathfrak{E}(m)$ выполняются шаги 6–11. Согласно устройству шагов 6–11 и определению графа $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2}$, верно $(c_{k+1}, \ell_{k+1}) \in X$ и у итерации $\mathfrak{E}(m)$ есть ребёнок $\mathfrak{I}_c$, для которого логическими условиями переходного набора являются c_{k+1} и ℓ_{k+1} и $[\mathfrak{I}_c] = v_{k+1}$. Получено противоречие с тем, что v_{k+1} по выбору k не является узлом ни одной итерации. ■

Алгоритм 4. Проверка эквивалентности программ

Вход: конечные программы $\pi_1 = (S_1, en_1, EX_1, T_1)$ и $\pi_2 = (S_2, en_2, EX_2, T_2)$.
Выход: ответ «Да», если $\pi_1 \sim_{\mathcal{F}} \pi_2$, и «Нет» иначе.

1: Вычислить значения $\pi_1' = \mathcal{A}_1(\pi_1)$, $\pi_2' = \mathcal{A}_1(\pi_2)$, $S_1^f = \mathcal{A}_2(\pi_1')$ и $S_2^f = \mathcal{A}_2(\pi_2')$, где $\mathcal{A}_1$ и $\mathcal{A}_2$ — алгоритмы 1 и 2 соответственно.
2: Вычислить и **вернуть** значение $\mathcal{A}_3(\pi_1', \pi_2', S_1^f, S_2^f)$, где $\mathcal{A}_3$ — алгоритм 3.

Теорема 1. Алгоритм 4 имеет сложность $O(\mathfrak{n}^2 f(\mathfrak{n}^2))$ для некоторой функции $f : \mathbb{N}_0 \to \mathbb{N}_0$, удовлетворяющей равенству $f(m) = \mathfrak{f}^{\mathcal{K}}(O(m))$, где $\mathfrak{n} = \max(|\pi_1|, |\pi_2|)$ для программ π_1 и π_2 на входе и $\mathfrak{f}^{\mathcal{K}}$ — сложностная характеристика алгоритмической составляющей системы $\mathcal{K}$, используемой в алгоритме 4.

Доказательство. По леммам 1 и 3, шаг 1 имеет сложность $O(\mathfrak{n})$. По устройству пополнения программы $|\pi_1'| = |\pi_1| + 1$ и $|\pi_2'| = |\pi_2| + 1$. Значит, на шаге 2 алгоритм 3 выполняется на программах размера не более $\mathfrak{n}+1$; по лемме 10 шаг 2 имеет сложность $O((\mathfrak{n}+1)^2 f(\mathfrak{n}^2)) = O(\mathfrak{n}^2 f(\mathfrak{n}^2))$ для некоторой f, для которой $f(m^2) = \mathfrak{f}^{\mathcal{K}}(O((m+1)^2)) = \mathfrak{f}^{\mathcal{K}}(O(m^2))$, а значит, $f(m) = \mathfrak{f}^{\mathcal{K}}(O(m))$. ■

Теорема 2. Для любых конечных программ π_1 и π_2 выполнение алгоритма 4 на входе (π_1, π_2) имеет результат «Да» тогда и только тогда, когда $\pi_1 \sim_{\mathcal{F}} \pi_2$.

Доказательство. Следует из устройства алгоритма, утверждения 4 и лемм 2, 4 и 11. ■

Пример 17. Рассмотрим программы π_1, π_2 из примера 1, шкалу $\mathcal{F}^{0,0}$ из примера 2, 1-ограниченную критериальную систему $\mathcal{K} = (\mathcal{W}, \varphi)$ из примера 13 и алгоритмическую составляющую этой системы со сложностной характеристикой $O(1)$, как отмечено в примере 14. Тогда алгоритм проверки эквивалентности (алгоритм 4) выполняется следующим образом. На шаге 1 вычисляются программы $\pi_1' = \pi_1^{\mathrm{loop}',a}$ и

$\pi_2' = \pi_2^{\mathrm{loop},a}$ и множества $S_1^f = \{\mathrm{loop}'\}$ и $S_2^f = \{\mathrm{loop}\}$ (программа π_2' представлена на рис. 7). На шаге 2 выполнение алгоритма 3 представляет собой обход графа $\Gamma^{\mathcal{K}}_{\pi_1',\pi_2'}$ в глубину [27] с произвольным выбором порядка исходящих дуг. Состояние loop' недостижимо из входа в π_1', поэтому обходятся только узлы графа $\Gamma^{\mathcal{K}}_{\pi_1,\pi_2'}$ (фрагмент этого графа представлен на рис. 12), и условие 1а алгоритма 3 не выполняется ни для одного посещённого узла. Если узел посещается больше одного раза, то по условию 1б алгоритма 3 дуги, исходящие из этого узла, не исследуются, что соответствует обходу в глубину. Если посещается опровергающий узел (как, например, отмеченные в примере 16), то по условию 3а алгоритма 3 алгоритм 4 завершается с ответом «Нет». Если посещаются узлы вида $((s,r),n_1)$ и $((s,r),n_2)$, где $n_1 \neq n_2$ (например, узлы $((s_3,\mathrm{loop}),0)$ и $((s_3,\mathrm{loop}),-1)$ на рис. 12), то по 1-ограниченности системы $\mathcal{K}$ и условию 3б алгоритма 3 алгоритм 4 завершается с ответом «Нет». Если обход графа завершается без ответа «Нет» по указанным причинам, то алгоритм 4 завершается с ответом «Да». В данном примере алгоритм обязательно завершается с ответом «Нет» в связи с наличием отмеченных выше узлов, этим обосновывается соотношение $\pi_1 \nsim_{\mathcal{F}^{0,0}} \pi_2$.

7. Применение алгоритма проверки эквивалентности программ

Рассмотрим $\mathfrak{A}$-моноид $\mathcal{M} = (M, \varepsilon, \circ)$. Записью $\mathcal{B}_{\mathcal{M}}$ обозначим подмоноид моноида $\mathcal{M} \times \mathcal{M}$ с множеством элементов $\{(\mathcal{M}(h), \mathcal{M}(g)) : h, g \in \mathfrak{A}^*,\ |h| = |g|\}$; записью $\mathcal{F}_{\mathcal{M}}$ — *шкалу моноида* $\mathcal{M}$, получающуюся из $\mathcal{M}$ заменой операции $\circ$ на её сужение на множество $M \times \mathfrak{A}$. *Моноидальной системой* для $\mathcal{M}$ назовём систему $\mathcal{K} = (\mathcal{W}, \mathcal{U}, w^+, w^*, \varphi)$, где $\mathcal{W} = (W, \epsilon, \odot)$ — конечно порождённый моноид; U — его подмоноид; $w^+, w^* \in \mathcal{W}$; φ — гомоморфизм моноида $\mathcal{B}_{\mathcal{M}}$ на $\mathcal{U}$ и для любой пары $(m_1, m_2) \in \mathcal{B}_{\mathcal{M}}$ справедлива равносильность $m_1 = m_2 \Leftrightarrow w^+ \odot \varphi(m_1, m_2) \odot w^* = \epsilon$. Такую систему $\mathcal{K}$ назовём $\mathfrak{k}$-*ограниченной*, если для любого элемента $w \in U \cdot w^*$ существует не более $\mathfrak{k}$ элементов $u \in w^+ \cdot U$, удовлетворяющих равенству $u \odot w = \epsilon$. *Производной системой* для $\mathcal{K}$ назовём пару $\mathcal{K}' = (\mathcal{W}', \psi)$, в которой $\mathcal{W}' = (W', w^+, \ominus)$, $W' = w^+ \odot U$, $\psi : \mathcal{B}_{\mathcal{F}_{\mathcal{M}}} \to W'$ задаётся равенством $\psi(m_1, m_2) = w^+ \odot \varphi(m_1, m_2)$ и $\ominus : W' \times (\mathfrak{A} \times \mathfrak{A}) \to W'$ — равенством $w \ominus (a, b) = w \odot \varphi(a, b)$.

Пример 18. Шкала $\mathcal{F}^{0,0}$ из примера 2 — это шкала свободного коммутативного моноида $\mathcal{M}$, порождённого множеством $\mathfrak{A} = \{a, b\}$. 1-Ограниченная моноидальная система $\mathcal{K} = (\mathcal{W}, \mathcal{U}, w^+, w^*, \varphi)$ для $\mathcal{M}$ может быть устроена так: $\mathcal{W} = \mathcal{U} = (\mathbb{Z}, 0, +)$; $w^+ = w^* = 0$; $\varphi(\mathcal{M}(h), \mathcal{M}(g)) = (n - m)$, где n и m — количество букв a в словах h и g соответственно. Критериальная система из примера 13 является производной системой для $\mathcal{K}$.

Лемма 12. Пусть $\mathcal{M}$ — $\mathfrak{A}$-моноид, $\mathcal{K}$ — его моноидальная система и $\mathcal{K}'$ — система, производная для $\mathcal{K}$. Тогда $\mathcal{K}'$ — это критериальная система шкалы $\mathcal{F}_{\mathcal{M}}$, и система $\mathcal{K}$ $\mathfrak{k}$-ограниченна тогда и только тогда, когда $\mathfrak{k}$-ограниченна система $\mathcal{K}'$.

Доказательство. Положим, что $\mathcal{M}$, $\mathcal{K}$ и $\mathcal{K}'$ имеют такой вид, как в определениях перед леммой, и $\mathcal{F}_{\mathcal{M}} = (M, \varepsilon, \cdot)$. Заметим, что $\mathcal{R}_{\mathcal{F}_{\mathcal{M}}} = M$ (так как $\mathcal{M}$ порождён множеством $\mathfrak{A}$) и множество элементов моноида $\mathcal{B}_{\mathcal{M}}$ есть $\mathcal{B}_{\mathcal{F}_{\mathcal{M}}}$. Тогда верно следующее (по соответствующим определениям):

1) $\psi(\varepsilon, \varepsilon) = w^+ \odot \varphi(\varepsilon, \varepsilon) = w^+ \odot \epsilon = w^+$;
2) для любых $m_1, m_2 \in \mathcal{M}$ имеет место $\psi(m_1, m_2) \ominus (a, b) = w^+ \odot \varphi(m_1, m_2) \odot \odot \varphi(a, b) = w^+ \odot \varphi(m_1 \circ a, m_1 \circ b) = \psi(m_1 \cdot a, m_2 \cdot b)$;
3) для любых $m_1, m_2 \in \mathcal{B}_{\mathcal{M}}$ если $w^+ \odot \varphi(m_1, m_2) \in \mathcal{E}_{\psi}$, то существует $m \in \mathcal{M}$, удовлетворяющий равенству $w^+ \odot \varphi(m_1, m_2) = \psi(m, m)$, т. е. $w^+ \odot \varphi(m_1, m_2) =$

$= w^+ \odot \varphi(m,m)$, а значит, $w^+ \odot \varphi(m_1,m_2) \odot w^* = w^+ \odot \varphi(m,m) \odot w^*$, при этом по определению моноидальной системы $w^+ \odot \varphi(m,m) \odot w^* = \epsilon$, а значит, $w^+ \odot \varphi(m_1,m_2) \odot w^* = \epsilon$; по тому же определению $m_1 = m_2$;

4) для любых $(m_1,m_2) \in \mathcal{B}_\mathcal{M}$ если $\psi(m_1,m_2) \in \mathcal{E}_\psi$, то $w^+ \odot \varphi(m_1,m_2) \in \mathcal{E}_\psi$ и по п. 3 верно $m_1 = m_2$.

Из пп. 1, 2 и 4 следует, что $\mathcal{K}'$ — критериальная система для $\mathcal{F}_\mathcal{M}$.

Положим, что система $\mathcal{K}'$ не $\mathfrak{k}$-ограниченна. Тогда существуют цепочки $h = a_1 \dots a_k$, $g = b_1 \dots b_k$ и попарно различные критерии $w_1, \dots, w_{\mathfrak{k}+1} \in W' = w^+ \odot U$, удовлетворяющие соотношению $\mathcal{W}'(w_i,h,g) \in \mathcal{E}_\psi$ для всех $i \in \{1,\dots,\mathfrak{k}+1\}$. При этом $\mathcal{W}'(w_i,h,g) = w_i \ominus (a_1,b_1) \ominus \dots \ominus (a_k,b_k) = w_i \odot \varphi(a_1,b_1) \odot \dots \odot \varphi(a_k,b_k) = w_i \odot \varphi(m_1,m_2)$ для $m_1 = \mathcal{M}(h)$ и $m_2 = \mathcal{M}(g)$. Соотношение $w_i \in w^+ \odot U$ означает, что w_i представим в виде $w^+ \odot \varphi(m_1^i, m_2^i)$ для некоторой пары $(m_1^i, m_2^i) \in \mathcal{B}_\mathcal{M}$. Значит, $w_i \odot \varphi(m_1,m_2) = w^+ \odot \varphi(m_1^i,m_2^i) \odot \varphi(m_1,m_2) = w^+ \odot \varphi(m_1^i \circ m_1, m_2^i \circ m_2) \in \mathcal{E}_\psi$, по доказанному п. 3 $m_1^i \circ m_1 = m_2^i \circ m_2$ и по определению моноидальной системы $w^+ \odot \varphi(m_1^i,m_2^i) \odot \varphi(m_1,m_2) \odot w^* = \epsilon$. То есть существуют $w' = w \odot w^* \in U \odot w^*$ и попарно различные $w_i \in w^+ \odot U$, $i \in \{1,\dots,\mathfrak{k}+1\}$, для которых $w_i \odot w' = \epsilon$. Следовательно, система $\mathcal{K}$ не $\mathfrak{k}$-ограниченна.

Положим, что система $\mathcal{K}$ не $\mathfrak{k}$-ограниченна. Тогда существуют $w \in U \odot w^*$ и попарно различные $w_1, \dots, w_{\mathfrak{k}+1} \in w^+ \odot U$, для которых $w_i \odot w = \epsilon$. По устройству классов $U \odot w^*$ и $w^+ \odot U$, $w = \varphi(m_1,m_2) \odot w^*$ для некоторой пары $(m_1,m_2) \in \mathcal{B}_\mathcal{M}$ и $w_i = w^+ \odot \varphi(m_1^i,m_2^i)$ для некоторой пары $(m_1^i,m_2^i) \in \mathcal{B}_\mathcal{M}$ для каждого $i \in \{1,\dots,\mathfrak{k}+1\}$. Тогда для каждого такого i верно следующее (по соответствующим определениям): $w^+ \odot \varphi(m_1^i \circ m_1, m_2^i \circ m_2) \odot w^* = w^+ \odot \varphi(m_1^i,m_2^i) \odot \varphi(m_1,m_2) \odot w^* = w_i \odot w = \epsilon$; $m_1^i \circ m_1 = m_2^i \circ m_2$ по определению моноидальной системы; $\psi(m_1^i \circ m_1, m_2^i \circ m_2) \in \mathcal{E}_\psi$ по утверждению 15; $m_1 = \mathcal{M}(h)$ и $m_2 = \mathcal{M}(g)$ для некоторых цепочек h, g одинаковой длины, так как $\mathfrak{A}$ — множество образующих моноида $\mathcal{M}$ и $(m_1,m_2) \in \mathcal{B}_\mathcal{M}$. Положим, что $h = a_1 \dots a_k$ и $g = b_1 \dots b_k$, где $k \in \mathbb{N}_0$ и $a_1, \dots, a_k, b_1, \dots, b_k \in \mathfrak{A}$; $\psi(m_1^i \circ m_1, m_2^i \circ m_2) = w^+ \odot \varphi(m_1^i \circ m_1, m_2^i \circ m_2) = w^+ \odot \varphi(m_1^i,m_2^i) \odot \varphi(a_1,b_1) \odot \dots \odot \varphi(a_k,b_k) = w_i \ominus (a_1,b_1) \ominus \dots \ominus (a_k,b_k) = \mathcal{W}'(w_i,h,g)$. Значит, $\mathcal{W}'(w_i,h,g) \in \mathcal{E}_\psi$, и так как это верно для всех $i \in \{1,\dots,\mathfrak{k}+1\}$, то система $\mathcal{K}'$ не является $\mathfrak{k}$-ограниченной. ■

Моноидом условной эквивалентности относительно множества $J \subseteq \mathfrak{A} \times \mathfrak{A} \times \mathfrak{A}$ назовём $\mathfrak{A}$-моноид с определяющими соотношениями $\{ca = cb : (a,b,c) \in J\}$. Из леммы 12, устройства и свойств моноидальных систем, описанных в [4, разд. 5] (там они называются критериальными системами), и устройства соответствующих производных систем следуют приведённые далее леммы 13–16.

Лемма 13. Для свободного $\mathfrak{A}$-моноида $\mathcal{M}$ существуют 1-ограниченная критериальная система $\mathcal{K}$ шкалы $\mathcal{F}_\mathcal{M}$ и её алгоритмическая составляющая со сложностной характеристикой $\mathfrak{f}^\mathcal{K}(n) = O(1)$.

Лемма 14. Для свободного коммутативного $\mathfrak{A}$-моноида $\mathcal{M}$ существуют 1-ограниченная критериальная система $\mathcal{K}$ шкалы $\mathcal{F}_\mathcal{M}$ и её алгоритмическая составляющая со сложностной характеристикой $\mathfrak{f}^\mathcal{K}(n) = O(1)$.

Лемма 15. Для любого частично коммутативного $\mathfrak{A}$-моноида $\mathcal{M}$ существуют 1-ограниченная критериальная система $\mathcal{K}$ шкалы $\mathcal{F}_\mathcal{M}$ и её алгоритмическая составляющая со сложностной характеристикой $\mathfrak{f}^\mathcal{K}(n) = O(n)$.

Лемма 16. Для любого моноида условной эквивалентности $\mathcal{M}$ существуют $2^{|\mathfrak{A}|}$-ограниченная критериальная система $\mathcal{K}$ шкалы $\mathcal{F}_\mathcal{M}$ и её алгоритмическая составляющая со сложностной характеристикой $\mathfrak{f}^\mathcal{K}(n) = O(1)$.

На основании этих лемм и алгоритма 4 можно получить соответствующие результаты о проверке эквивалентности программ, приведённые далее в теоремах 3–6. Прикладные причины рассмотрения шкал, для которых сформулированы эти теоремы, можно подробно изучить, например, в [4].

Теорема 3. Существует алгоритм проверки сильной эквивалентности пропозициональных программ Мили, имеющий сложность $O(n^2)$.

Доказательство. Заметим, что сильная эквивалентность программ равносильна их эквивалентности на шкале свободного моноида: необходимость — по определению сильной эквивалентности, достаточность сформулирована в [4, следствие 3]. Справедливость теоремы 3 следует из этого, теорем 1 и 2 и леммы 13. ■

Теорема 4. Для любого свободного коммутативного $\mathfrak{A}$-моноида существует алгоритм проверки эквивалентности пропозициональных программ Мили на шкале этого моноида, имеющий сложность $O(n^2)$.

Доказательство. Следует из теорем 1 и 2 и леммы 14. ■

Теорема 5. Для любого частично коммутативного $\mathfrak{A}$-моноида существует алгоритм проверки эквивалентности пропозициональных программ Мили на шкале этого моноида, имеющий сложность $O(n^4)$.

Доказательство. Следует из теорем 1 и 2 и леммы 15. ■

Теорема 6. Для любого моноида условной эквивалентности существует алгоритм проверки эквивалентности пропозициональных программ Мили на шкале этого моноида, имеющий сложность $O(n^2)$.

Доказательство. Следует из теорем 1 и 2 и леммы 16. ■

Заключение

Как отмечалось во введении, соотношение между ПППЗ и ППМ схоже с соотношением между автоматами Мура и Мили, то есть модель ППМ можно считать в некотором роде более общей по сравнению с ПППЗ. Но всё же эти модели, вообще говоря, несравнимы по тем же причинам, отмеченным во введении, по которым несравнимы модели ПППЗ и дискретных преобразователей Глушкова — Летичевского. Исследование соотношения между этими моделями оставлено на будущее.

Ключевые результаты данной работы — это переложение результатов [4] с ПППЗ на ППМ: техники совместных вычислений (понятие критериального графа, алгоритм 4 и теоремы 1 и 2) и эффективных алгоритмов проверки эквивалентности ППМ на некоторых полезных шкалах, получающихся применением этой техники (теоремы 3–6). Ещё один результат, представляющий интерес, — это лемма 12, позволяющая при получении алгоритмов проверки эквивалентности ПППЗ на основе техники из [4] немедленно в качестве следствия получать настолько же эффективные аналогичные алгоритмы проверки эквивалентности ППМ. Кроме того, можно выделить ещё несколько особенностей полученных результатов, показывающих их ценность.

В работах, использующих технику совместных вычислений с критериальными системами [4, 8–11, 14, 20, 21, 31–37], рассматриваются только шкалы $\mathfrak{A}$-моноидов, имеющие критериальные системы, основывающиеся на конечно порождённых моноидах аналогично тому, как в данной работе задание моноидальной системы начинается с такого моноида. В работе показано, что можно применять эту технику и к шкалам, не базирующимся на моноидах (согласно теоремам 1 и 2), и не основывать понятие критериальной системы на моноидах. В лемме 12 показано, что понятие критериальной

системы из [4], опирающееся на моноиды, является частным случаем понятия критериальной системы, введённого в данной работе.

Техника совместных вычислений в данной работе по сравнению с [4] заметно приближена к технике проверки эквивалентности конечных автоматов с помощью их декартова произведения [22, 38], что выражается, в числе прочего, в использовании в ключевых определениях операций $\oplus$ и $\otimes$, являющихся по сути разновидностями декартова произведения вычислителей автоматного типа.

Кроме того, в работе исправлен ряд огрехов, содержащихся в [4] и проявляющихся в остальных работах, посвящённых ПППЗ и технике совместных вычислений:

1. Явно указан способ подсчёта сложности алгоритмов, включая модель сложности и способы представления данных. Для результатов, констатирующих или опровергающих полиномиальную разрешимость, это было бы неважно, но когда речь идёт о более точных оценках сложности, это становится важным.
2. Рассуждения о сравнении преобразователей за логарифмическое время в доказательстве теоремы 7 работы [4] склоняют к тому, что в [4] для подсчёта сложности используется модель машин Тьюринга или родственная ей. В данной работе вместо неё используется более широко применяющаяся на практике модель RAM-машин и в связи с этим получаются оценки сложности, более близкие к практике.
3. Алгоритмы снабжены всеми подробностями, необходимыми для анализа и подсчёта сложности, и не содержат существенных недосказанностей, которые в [4] приводят, например, к тому, что:
   - в формулировке теоремы 7 используется сложность $\mathfrak{f}^{\mathcal{K}}_{=}$, а следовало бы использовать $\mathfrak{f}^{\mathcal{K}}$ (если применить обозначения данной работы по аналогии);
   - после изучения доказательства теоремы 7 остаётся сомнение, не потерян ли в оценке сложности какой-либо дополнительный множитель, проистекающий из копирования данных и особенностей работы со структурами данных (достоверный вывод, что не потерян, можно сделать только после дополнительного не очень тривиального анализа обоснования);
   - в теореме 10 без достаточных пояснений приводится оценка $O(n^3 \log n)$, тогда как, согласно лемме 15 данной работы, разумно было бы предположить оценку не лучше чем $O(n^4)$ (найти, откуда следовала бы оценка $O(n^3 \log n)$, не удалось).

В будущем планируется: а) обосновать, что модель ППМ можно считать обобщением модели ПППЗ; б) усовершенствовать технику совместных вычислений для получения более низкого порядка сложности соответствующих алгоритмов проверки эквивалентности; в) применить полученные наработки для аналогичного усовершенствования известных смежных результатов и затем для установления новых фактов об эффективной разрешимости проблемы эквивалентности в моделях программ.

ЛИТЕРАТУРА

1. *Rice H. G.* Classes of recursively enumerable sets and their decision problems // Trans. AMS. 1953. V. 74. P. 358–366.
2. *Клини С. К.* Введение в метаматематику. М.: ИЛ, 1957.
3. *Глушков В. М., Летичевский А. А.* Теория дискретных преобразователей // Избранные вопросы алгебры и логики. Новосибирск: Наука, Сибирское отделение, 1973. С. 5–39.

4. *Захаров В. А.* Быстрые алгоритмы разрешения эквивалентности операторных программ на уравновешенных шкалах // Матем. вопр. кибернетики. 1998. Вып. 7. С. 303–324.
5. *Летичевский А. А.* Функциональная эквивалентность дискретных преобразователей III // Кибернетика. 1972. № 1. С. 1–4.
6. *Летичевский А. А., Смикун Л. Б.* Об одном классе групп с разрешимой проблемой эквивалентности // Докл. АН СССР. 1976. Т. 227. № 1. С. 36–38.
7. *Захаров В. А., Подымов В. В.* Применение алгоритмов проверки эквивалентности для оптимизации программ // Труды ИСП РАН. 2015. Т. 27. Вып. 4. С. 145–174.
8. *Zakharov V. A.* An efficient and unified approach to the decidability of equivalence of propositional programs // LNCS. 1998. V. 1443. P. 247–258.
9. *Захаров В. А.* Быстрые алгоритмы разрешения эквивалентности пропозициональных операторных программ на упорядоченных полугрупповых шкалах // Вестн. Моск. ун-та. Сер. 15. Вычислительная математика и кибернетика. 1999. № 3. С. 29–35.
10. *Захаров В. А.* О проблеме эквивалентности операторных программ на уравновешенных однородных обратимых шкалах // Матем. вопр. кибернетики. 2001. Вып. 10. С. 155–166.
11. *Zakharov V. A.* The equivalence problem for computational models: Decidable and undecidable cases // LNCS. 2001. V. 2055. P. 133–152.
12. *Zakharyaschev I. M. and Zakharov V. A.* On the equivalence-checking problem for polysemantic models of sequential programs // Труды ИСП РАН. 2004. Т. 6. С. 179–198.
13. *Podlovchenko R. I., Rusakov D. M., and Zakharov V. A.* The equivalence problem for programs with mode switching is PSPACE-complete // Труды ИСП РАН. 2006. Т. 11. С. 109–128.
14. *Щербина В. Л., Захаров В. А.* Эффективные алгоритмы проверки эквивалентности программ в моделях, связанных с обработкой прерываний // Вестн. Моск. ун-та. Сер. 15. Вычислительная математика и кибернетика. 2008. № 2. С. 33–41.
15. *Подловченко Р. И., Кузюрин Н. Н., Щербина В. Л., Захаров В. А.* Использование алгебраических моделей программ для обнаружения метаморфного вредоносного кода // Фундамент. и прикл. матем. 2009. Т. 15. № 5. С. 181–198.
16. *Zakharov V. A.* Program equivalence checking by two-tape automata // Cybernetics and Systems Analysis. 2010. V. 46. No. 4. P. 554–562.
17. *Подымов В. В., Захаров В. А.* О двухленточных машинах, описывающих полугруппы с сокращением // Проблемы теоретической кибернетики. Материалы XVI Междунар. конф. (Нижний Новгород, 20–25 июня 2011 г.). Н. Новгород: Изд-во Нижегор. ун-та, 2011. С. 372–375.
18. *Захаров В. А.* Модели и алгоритмы в задаче проверки эквивалентности программ // Материалы XI Междунар. семинара «Дискретная математика и ее приложения», посвященного 80-летию со дня рождения академика О. Б. Лупанова (Москва, МГУ, 18–23 июня 2012 г.), М.: Изд-во механико-математического факультета МГУ, 2012. С. 53–62.
19. *Подловченко Р. И., Захаров В. А.* О двух методах распознавания эквивалентности в алгебраических моделях программ // Интеллектуальные системы. 2013. Т. 17. № 1–4. С. 366–370.
20. *Подымов В. В., Захаров В. А.* Полиномиальный алгоритм проверки эквивалентности в модели программ с перестановочными и подавляемыми операторами // Труды ИСП РАН. 2014. Т. 26. Вып. 3. С. 145–166.
21. *Подымов В. В.* Улучшение алгоритмов проверки эквивалентности операторных программ при помощи анализа весов вершин // Ломоносовские чтения-2021. Секция Вычислительной математики и кибернетики. М.: Изд-во Моск. ун-та, 2021. С. 124–125.
22. *Карпов Ю. Г.* Теория автоматов. СПб.: Питер, 2003.

23. *Zakharov V. A. and Zakharyaschev I. M.* On the equivalence checking problem for a model of programs related with muti-tape automata // LNCS. 2005. V. 3317. P. 293–305.
24. *Пентус А. Е., Пентус М. Р.* Теория формальных языков: учеб. пособие. М.: Изд-во ЦПИ при механико-математическом факультете МГУ, 2004.
25. *Гаврилов Г. П., Сапоженко А. А.* Задачи и упражнения по дискретной математике: учеб. пособие. 3-е изд., перераб. М.: Физматлит, 2005.
26. *Зорич В. А.* Математический анализ. Ч. I. 4-е изд., испр. М.: МЦНМО, 2002.
27. *Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К.* Алгоритмы: построение и анализ. 3-е изд. М.: ООО «И. Д. Вильямс», 2013.
28. *Лаллеман Ж.* Полугруппы и комбинаторные приложения. М.: Мир, 1985.
29. *Клиффорд А., Престон Г.* Алгебраическая теория полугрупп. Т. 1. М.: Мир, 1972.
30. *Страуструп Б.* Язык программирования C++. Краткий курс. 2-е изд. СПб.: ООО «Диалектика», 2019.
31. *Захаров В. А.* Об эффективной разрешимости проблемы эквивалентности линейных унарных рекурсивных программ // Матем. вопр. кибернетики. 1999. Т. 8. С. 255–273.
32. *Захаров В. А.* Об эквивалентности потоковых программ // Материалы XI Междунар. семинара «Дискретная математика и ее приложения», посвященного 80-летию со дня рождения академика О. Б. Лупанова (Москва, МГУ, 18–23 июня 2012 г.), М.: Изд-во механико-математического факультета МГУ, 2012. С. 119–122.
33. *Подымов В. В.* Алгоритм проверки эквивалентности линейных унарных рекурсивных программ на упорядоченных полугрупповых шкалах // Вестн. Моск. ун-та. Сер. 15. Вычислительная математика и кибернетика. 2012. № 4. С. 37–43.
34. *Захаров В. А.* Об эквивалентности ограниченно недетерминированных автоматов-преобразователей над полугруппами // Проблемы теоретической кибернетики. Материалы XVII Междунар. конф. (Казань, 16–20 июня 2014 г.). Казань: Отечество, 2014. С. 100–102.
35. *Захаров В. А.* Моделирование и анализ поведения последовательных реагирующих программ // Труды ИСП РАН. 2015. Т. 27. № 2. С. 221–250.
36. *Подымов В. В.* Алгоритмы проверки эквивалентности программ с процедурами в прогрессивных полугрупповых перегородчатых моделях // Вестн. Моск. ун-та. Сер. 15. Вычислительная математика и кибернетика. 2019. № 4. С. 37–44.
37. *Подымов В. В.* О сложности проверки эквивалентности линейных унарных рекурсивных программ над уравновешенными полугруппами // Ломоносовские чтения-2024. Секция Вычислительной математики и кибернетики. М.: МАКС Пресс, 2024. С. 53–55.
38. *Котов В. Е., Сабельфельд В. К.* Теория схем программ. М.: Наука, 1991.

REFERENCES

1. *Rice H. G.* Classes of recursively enumerable sets and their decision problems. Trans. AMS, 1953, vol. 74, pp. 358–366.
2. *Kleene S. C.* Introduction to Metamathematics. N.Y., Toronto, Nostrand Company, 1952.
3. *Glushkov V. M. and Letichevskiy A. A.* Teoriya diskretnykh preobrazovateley [Theory of discrete processors]. Izbrannye Voprosy Algebry i Logiki, Novosibirsk, Nauka, 1973, pp. 5–39. (in Russian)
4. *Zakharov V. A.* Bystrye algoritmy razresheniya ekvivalentnosti operatornykh programm na uravnoveshennykh shkalakh [Fast equivalence-checking algorithms for operator programs over balanced frames]. Matematicheskie Voprosy Kibernetiki, 1998, iss. 7, pp. 303–324. (in Russian)
5. *Letichevskiy A. A.* Funktsional'naya ekvivalentnost' diskretnykh preobrazovateley III [Functional equivalence of discrete processors III]. Kibernetika, 1972, no. 1, pp. 1–4. (in Russian)

6. *Letichevskiy A. A. and Smikun L. B.* Ob odnom klasse grupp s razreshimoy problemoy ekvivalentnosti [On a class of groups with solvable problem of automata equivalence]. Doklady AN SSSR, 1976, vol. 227, no. 1, pp. 36–38. (in Russian)
7. *Zakharov V. A. and Podymov V. V.* Primenenie algoritmov proverki ekvivalentnosti dlya optimizatsii programm [On the application of equivalence checking algorithms for program minimization]. Proc. ISP RAS, 2015, vol. 27, iss. 4, pp. 145–174. (in Russian)
8. *Zakharov V. A.* An efficient and unified approach to the decidability of equivalence of propositional programs. LNCS, 1998, vol. 1443, pp. 247–258.
9. *Zakharov V. A.* Bystrye algoritmy razresheniya ekvivalentnosti propozitsional'nykh operatornykh programm na uporyadochennykh polugruppovykh shkalakh [Fast equivalence-checking algorithms for operator programs over ordered frames]. Vestnik Moskovskogo Universiteta. Ser. 15. Vychislitel'naya matematika i kibernetika, 1999, no. 3, pp. 29–35. (in Russian)
10. *Zakharov V. A.* O probleme ekvivalentnosti operatornykh programm na uravnoveshennykh odnorodnykh obratimykh shkalakh [On the equivalence problem for operator programs over balanced homogenous invertible frames]. Matematicheskie Voprosy Kibernetiki, 2001, iss. 10, pp. 155–166. (in Russian)
11. *Zakharov V. A.* The equivalence problem for computational models: Decidable and undecidable cases. LNCS, 2001, vol. 2055, pp. 133–152.
12. *Zakharyaschev I. M. and Zakharov V. A.* On the equivalence-checking problem for polysemantic models of sequential programs. Proc. ISP RAS, 2004, vol. 6, pp. 179–198.
13. *Podlovchenko R. I., Rusakov D. M., and Zakharov V. A.* The equivalence problem for programs with mode switching is PSPACE-complete. Proc. ISP RAS, 2006, vol. 11, pp. 109–128.
14. *Shcherbina V. L. and Zakharov V. A.* Effektivnye algoritmy proverki ekvivalentnosti programm v modelyakh, svyazannykh s obrabotkoy preryvaniy [Efficient equivalence-checking algorithms for program models related to interrupt handling]. Vestnik Moskovskogo Universiteta. Ser. 15. Vychislitel'naya matematika i kibernetika, 2008, no. 2, pp. 33–41. (in Russian)
15. *Podlovchenko R. I., Kuzyurin N. N., Shcherbina V. L., and Zakharov V. A.* Using algebraic models of programs for detecting metamorphic malwares. J. Math. Sci., 2011, vol. 172, no. 5, pp. 740–750.
16. *Zakharov V. A.* Program equivalence checking by two-tape automata. Cybernetics and Systems Analysis, 2010, vol. 46, no. 4, pp. 554–562.
17. *Podymov V. V. and Zakharov V. A.* O dvukhlentochnykh mashinakh, opisyvayushchikh polugruppy s sokrashcheniem [On two-tape machines used for description of cancellative semigroups]. Proc. XVI Intern. Conf. "Problemy Teoreticheskoy Kibernetiki" (Nizhny Novgorod, June 20–25, 2011), Nizhniy Novgorod, UNN Publ., 2011, pp. 372–375. (in Russian)
18. *Zakharov V. A.* Modeli i algoritmy v zadache proverki ekvivalentnosti programm [Models and algorithms related to program equivalence checking]. Proc. XI Intern. Conf. "Diskretnaya Matematika i ee Prilozheniya" (Moscow, MSU, June 18–23, 2012), Moscow, MSU Faculty of Mechanics and Mathematics Publ., 2012, pp. 53–62. (in Russian)
19. *Podlovchenko R. I. and Zakharov V. A.* O dvukh metodakh raspoznavaniya ekvivalentnosti v algebraicheskikh modelyakh programm [On two equivalence-checking techniques for algebraic program models]. Intellektual'nye Sistemy, 2013, vol. 17, no. 1–4, pp. 366–370. (in Russian)
20. *Podymov V. V. and Zakharov V. A.* Polinomial'nyy algoritm proverki ekvivalentnosti v modeli programm s perestanovochnymi i podavlyaemymi operatorami [A polynomial algorithm for checking the equivalence in models of programs with commutation and vast operators]. Proc. ISP RAS, 2014, vol. 26, iss. 3, pp. 145–166. (in Russian)
21. *Podymov V. V.* Uluchshenie algoritmov proverki ekvivalentnosti operatornykh programm pri pomoshchi analiza vesov vershin [Improving equivalence-checking algorithms for operator

programs with node weight analysis]. Lomonosovskie Chteniya-2021, Moscow, MSU Publ., 2021, pp. 124–125. (in Russian)

22. *Karpov Yu. G.* Teoriya avtomatov [Automata Theory]. Saint Petersburg, Piter, 2003. (in Russian)
23. *Zakharov V. A. and Zakharyaschev I. M.* On the equivalence checking problem for a model of programs related with muti-tape automata. LNCS, 2005, vol. 3317, pp. 293–305.
24. *Pentus A. E. and Pentus M. R.* Teoriya formal'nykh yazykov [Formal Language Theory]. Moscow, MSU Faculty of Mechanics and Mathematics Publ., 2004. (in Russian)
25. *Gavrilov G. P. and Sapozhenko A. A.* Zadachi i uprazhneniya po diskretnoy matematike [Discrete Mathematics Problems and Exercises]. Moscow, Fizmatlit Publ., 2005. (in Russian)
26. *Zorich V. A.* Matematicheskiy analiz [Mathematical Analysis]. Part I. Moscow, MCCME Publ., 2002. (in Russian)
27. *Cormen T. H., Leiserson C. E., Rivest R. L., and Stein C.* Introduction to Algorithms. 3rd ed. Cambridge, Massachusetts, London, England, MIT Press, 2009.
28. *Lalleman G.* Semigroups and Combinatorial Applications. N.Y., Chichester, Brisbane, Toronto, John Wiley & Sons, 1979.
29. *Clifford A. H. and Preston G. B.* The Algebraic Theory of Semigroups, vol. I. Providence, Rhode Island, AMS, 1964.
30. *Stroustrup B.* A Tour of C++. 2nd ed. Boston, MA, USA, Addison-Wesley, 2018.
31. *Zakharov V. A.* Ob effektivnoy razreshimosti problemy ekvivalentnosti lineynykh unarnykh rekursivnykh programm [On efficient decidability of an equivalence problem for linear monadic recursive programs]. Matematicheskie Vopposy Kibepnetiki, 1999, vol. 8, pp. 255–273. (in Russian)
32. *Zakharov V. A.* Ob ekvivalentnosti potokovykh programm [On equivalence of streaming programs]. Proc. XI Intern. Conf. "Diskretnaya Matematika i ee Prilozheniya" (Moscow, June 18–23, 2012), Moscow, MSU Faculty of Mechanics and Mathematics Publ., 2012, pp. 119–122. (in Russian)
33. *Podymov V. V.* Algoritm proverki ekvivalentnosti lineynykh unarnykh rekursivnykh programm na uporyadochennykh polugruppovykh shkalakh [An equivalence-checking algorithm for linear monadic recursive programs over ordered semigroup frames]. Vestnik Moskovskogo universiteta. Ser. 15. Vychislitel'naya Matematika i Kibernetika, 2012, no. 4, pp. 37–43. (in Russian)
34. *Zakharov V. A.* Ob ekvivalentnosti ogranichenno nedeterminirovannykh avtomatov-preobrazovateley nad polugruppami [On equivalence of finite-valued transducers over semigroups]. Proc. XVII Intern. Conf. "Problemy Teoreticheskoy Kibernetiki" (Kazan, June 16–20, 2014), Kazan, Otechestvo Publ., 2014, pp. 100–102. (in Russian)
35. *Zakharov V. A.* Modelirovanie i analiz povedeniya posledovatel'nykh reagiruyushchikh programm [Modeling and analysis of the behavior of successive reactive programs]. Proc. ISP RAS, 2015, vol. 27, no. 2, pp. 221–250. (in Russian)
36. *Podymov V. V.* Efficient equivalence-checking algorithms for procedural programs in progressive semigroup gateway models. Moscow Univ. Comput. Math. Cybern., 2019, vol. 43, no. 4, pp. 181–187.
37. *Podymov V. V.* O slozhnosti proverki ekvivalentnosti lineynykh unarnykh rekursivnykh programm nad uravnoveshennymi polugruppami [On complexity of equivalence checking for linear monadic recursive programs over balanced semigroups]. Lomonosovskie Chteniya-2024, Moscow, MAKS Press, 2024, pp. 53–55. (in Russian)
38. *Kotov V. E. and Sabelfeld V. K.* Teoriya skhem programm [Theory of Program Schemes]. Moscow, Nauka, 1991. (in Russian)

УДК 510.52 DOI 10.17223/20710410/70/6

О ГЕНЕРИЧЕСКОЙ СЛОЖНОСТИ РЕШЕНИЯ УРАВНЕНИЙ В КОНЕЧНЫХ ПРЕДИКАТНЫХ АЛГЕБРАИЧЕСКИХ СИСТЕМАХ[1]

А. Н. Рыбалов

Институт математики им. С. Л. Соболева СО РАН, г. Омск, Россия

E-mail: alexander.rybalov@gmail.com

Изучается генерическая сложность двух вариантов проблемы решения уравнений без констант над конечными предикатными алгебраическими системами: распознавания разрешимости и поиска решения. Для обеих проблем во многих случаях неизвестно эффективных полиномиальных алгоритмов. Предлагается полиномиальный генерический алгоритм для проблемы распознавания разрешимости. С другой стороны, для проблемы поиска решения доказывается, что если для неё нет полиномиального вероятностного алгоритма, то существует подпроблема этой проблемы, для которой нет полиномиального генерического алгоритма. Полученный результат является теоретическим обоснованием возможных приложений проблемы поиска решения в криптографии, где нужно, чтобы проблема взлома криптоалгоритма была трудной для почти всех входов.

Ключевые слова: *генерическая сложность, конечные алгебраические системы, уравнения.*

ON GENERIC COMPLEXITY OF SOLVING OF EQUATIONS IN FINITE PREDICATE ALGEBRAIC STRUCTURES

A. N. Rybalov

Sobolev Institute of Mathematics, Omsk, Russia

In this paper, we study the generic complexity of two variants of the problem of solving equations without constants over finite predicate algebraic systems: the solvability recognition problem and the solution search problem. For both problems, efficient polynomial algorithms are not known in many cases. We propose a polynomial generic algorithm for the solvability recognition problem. On the other hand, for the solution search problem, we prove that if there is no polynomial probabilistic algorithm for it, then there is a subproblem of this problem for which there is no polynomial generic algorithm. The obtained result is a theoretical justification for possible applications of the solution search problem in cryptography, where the problem of breaking a cryptographic algorithm is required to be hard for almost all inputs. To prove this theorem, we use the method of generic amplification, which allows to construct generically hard problems from the problems hard in the classical sense. The main ingredient of this method is a technique of cloning, which unites inputs of the problem together in the large enough sets of equivalent inputs. Equivalence is understood in the sense that the problem is solved similarly for them.

Keywords: *generic complexity, finite algebraic structures, equations.*

[1]Работа поддержана грантом Российского научного фонда № 25-11-20023.

Введение

Решение уравнений и систем уравнений над вещественными, комплексными, рациональными, целыми числами является классической темой исследований в различных областях математики в течение тысяч лет. В последние десятилетия фокус исследований перемещается на неклассические области, такие, как группы [1], полугруппы [2–5], графы [6], частичные порядки [7]. Потребность решения уравнений в этих системах возникает при рассмотрении различных практических проблем информатики, криптографии, теории языков программирования. Например, свободные полугруппы являются базисом для описания важнейших классов формальных языков и грамматик: регулярных, контекстно свободных. Часто при этом изучаемый формальный язык задаётся некоторым набором уравнений, множество решений которых даёт нужный язык. К необходимости решения уравнений над графами приводят задачи проверки вложимости (совместимости) одной коммуникационной сети в другую.

Особый интерес представляет изучение вычислительной сложности проблемы решения уравнений над конечными алгебраическими системами. Очень часто здесь возникает так называемая дихотомия: для каких-то конечных систем данного класса эта проблема разрешима за полиномиальное время, для всех других является NP-полной. Это явление характерно для классов конечных групп [8], конечных полугрупп [9], конечных графов [6] (для систем уравнений без констант). Например, для конечных графов дихотомия зависит от хроматического числа графа, над которым решаются уравнения: если оно не превосходит двух, то проблема разрешима за полиномиальное время, иначе — NP-полна. Напомним, что хроматическое число графа — это минимальное число цветов, в которые можно раскрасить вершины так, чтобы любые вершины, соединённые ребром, были покрашены в разные цвета. Этот результат [6] был получен для систем уравнений без констант, однако аналогичный результат для систем с константами легко следует из работ [6, 10].

NP-полнота позволяет эффективно сводить другие практически важные NP-полные проблемы к проблеме решения уравнений и использовать мощные алгебраические методы для разработки более эффективных алгоритмов их решения. Кроме того, в случае NP-полноты проблемы решения уравнений актуальным является изучение её генерической сложности [11]. В рамках генерического подхода алгоритмическая проблема рассматривается не на всём множестве входов, а на некотором подмножестве «почти всех» входов. С одной стороны, положительные результаты о возможности эффективного решения каких-либо трудных задач для почти всех входов полезны для практики. С другой стороны, негативные результаты о генерической трудности некоторых проблем дают надежду на возможное их использование в криптографии, где как раз важно, чтобы проблема взлома криптосистемы была трудной для почти всех входов. Генерическая сложность проблем решения уравнений над конечными полями и полугруппами рассмотрена в [12].

В данной работе изучается генерическая сложность двух вариантов проблемы решения уравнений без констант в конечных предикатных алгебраических системах. Первый вариант — проблема распознавания разрешимости систем уравнений. Здесь входом является произвольная система уравнений без констант, необходимо определить, существует ли у неё решение. Второй вариант — проблема поиска решения системы уравнений. Для этой проблемы входом является система уравнений без констант, для которой заведомо существует решение, нужно найти хотя бы одно её решение. Проблемы поиска, в отличие от проблем распознавания, находят применения в криптографии, где всегда известно, что решение есть и надо его найти. В работе

предлагается полиномиальный генерический алгоритм для проблемы распознавания разрешимости систем уравнений. С другой стороны, для проблемы поиска решения доказывается, что если для неё не существует полиномиального вероятностного алгоритма, то существует подпроблема этой проблемы, для которой нет полиномиального генерического алгоритма. Вероятностные алгоритмы в процессе своей работы могут использовать датчик случайных чисел, что позволяет ускорять вычисления. Однако считается, что любой полиномиальный вероятностный алгоритм можно эффективно дерандомизировать, построив полиномиальный алгоритм, не использующий генератор случайных чисел и решающий ту же самую проблему. Хотя этот факт до сих пор не доказан, имеются веские основания в пользу него [13].

1. Предварительные сведения

На протяжении всей работы будем рассматривать системы уравнений без констант. Пусть $\mathfrak{A} = \langle A, \sigma \rangle$ — алгебраическая система с предикатной сигнатурой $\sigma = \{P_i^{(k_i)} : i = 1, \ldots, m\}$. *Уравнением* над $\mathfrak{A}$ называется формула одного из двух типов:

1) $(x_i = x_j)$;
2) $P_i(x_1, \ldots, x_{k_i})$, $P_i \in \sigma$, $i = 1, \ldots, m$.

Системой уравнений над $\mathfrak{A}$ называется конечный набор уравнений. *Решение* системы уравнений S от переменных $x_1, \ldots, x_t$ — это такой набор $a_1, \ldots, a_t$ элементов из A, который при подстановке в каждое уравнение системы S даёт истинную над $\mathfrak{A}$ формулу. Легко видеть, что для любой системы S над $\mathfrak{A}$ существует эквивалентная ей система S', в которой отсутствуют уравнения вида $(x_i = x_j)$. Действительно, для удаления таких уравнений достаточно во всех остальных уравнениях заменить переменную x_j на переменную x_i. Поэтому в дальнейшем будем рассматривать системы, в которых все уравнения имеют тип 2, то есть являются предикатами от переменных.

Проблема распознавания разрешимости систем уравнений над $\mathfrak{A}$ формулируется следующим образом. По произвольной заданной системе уравнений S определить, существует ли у неё решение в $\mathfrak{A}$. *Проблема поиска решения* систем уравнений над $\mathfrak{A}$ формулируется немного иначе. По произвольной заданной разрешимой системе уравнений S найти хотя бы одно её решение в $\mathfrak{A}$.

Напомним основные определения генерического подхода [11]. Пусть I — некоторое множество входов, а I_n — подмножество входов размера n. Для подмножества $S_n \subseteq I$ определим последовательность

$$\rho_n(S) = \frac{|S_n|}{|I_n|}, \quad n = 1, 2, 3, \ldots,$$

где $S_n = S \cap I_n$ — множество входов из S размера n. *Асимптотической плотностью* S назовём предел

$$\rho(S) = \lim_{n \to \infty} \rho_n(S).$$

Множество S называется *пренебрежимым*, если его асимптотическая плотность $\rho(S) = 0$.

Алгоритм $\mathcal{A}$ с множеством входов I и множеством выходов $J \cup \{\square\}$ $(\square \notin J)$ называется *генерическим*, если

1) $\mathcal{A}$ останавливается на всех входах из I;
2) множество $\{x \in I : \mathcal{A}(x) = \square\}$ является пренебрежимым.

Здесь символ $\square$ обозначает неопределённый ответ. Генерический алгоритм $\mathcal{A}$ *вычисляет* функцию $f : I \to \mathbb{N}$, если для всех $x \in I$ выполнено

$$(\mathcal{A}(x) \neq \square) \Rightarrow (f(x) = \mathcal{A}(x)).$$

Проблема распознавания множества $A \subseteq I$ *генерически разрешима за полиномиальное время*, если существует полиномиальный генерический алгоритм, вычисляющий характеристическую функцию множества A. Напомним, что *характеристической функцией* множества $A \subseteq I$ называется функция $\chi_A : I \to \{0, 1\}$, определённая следующим образом:

$$\chi_A(x) = \begin{cases} 1, & \text{если } x \in A, \\ 0, & \text{если } x \notin A. \end{cases}$$

Напомним также некоторые понятия классической теории сложности вычислений [14]. *Время работы* $t_M(x)$ машины Тьюринга M на входе $x \in I$ — это число шагов машины от начала работы до остановки. Машина Тьюринга M *полиномиальна*, если существует полином $p(n)$, такой, что для любого $x \in I$ имеет место $t_M(x) < p(\text{size}(x))$.

Вероятностная машина Тьюринга — это машина Тьюринга, в программе которой допускаются пары недетерминированных правил, которые одновременно применимы в данной ситуации. В процессе работы такой машины с вероятностью $1/2$ выбирается первое правило и с вероятностью $1/2$ — второе. Время работы $t_M(x, \tau)$ вероятностной машины Тьюринга на входе x зависит от вычислительного пути (последовательности выполненных команд) τ. Вероятностная машина Тьюринга M называется *полиномиальной*, если существует полином $p(n)$, такой, что для любого x и для любого вычислительного пути τ машины M на x имеет место $t_M(x, \tau) < p(\text{size}(x))$.

Обозначим через $\Pr[M(x) = y]$ вероятность того, что машина M на входе x выдаёт ответ y. Вероятностная машина M *вычисляет* функцию $f : I \to J$, если для любого $x \in I$ имеет место

$$\big(f(x) = y\big) \ \Rightarrow \Pr[M(x) = y] > 2/3.$$

Вероятностные машины Тьюринга формализуют понятие алгоритма, использующего генератор случайных чисел.

2. Генерический алгоритм распознавания разрешимости систем уравнений

Пусть $\mathfrak{A} = \langle A, \sigma \rangle$ — конечная алгебраическая система с предикатной сигнатурой $\sigma = \{P_i^{(k_i)} : i = 1, \ldots, m\}$. Будем представлять системы уравнений над $\mathfrak{A}$ следующим образом. Во-первых, зафиксируем переменные системы $x_1, \ldots, x_n$. Число переменных n — *размер системы*. По каждому предикату $P_i^{(k_i)}$, $i = 1, \ldots, m$, из сигнатуры σ рассмотрим так называемую *таблицу включения* — это k_i-мерный куб с n позициями по каждой размерности. Итого получается n^{k_i} мест. На месте с координатами $(j_1, \ldots, j_{k_i})$ записываем 1, если в системе есть уравнение $P_i(x_{j_1}, \ldots, x_{j_{k_i}})$, и 0, если нет. *Представлением* системы уравнений является набор таблиц включения для каждого предиката сигнатуры, встречающегося в этой системе. Обозначим через $\mathcal{S}$ множество систем уравнений над $\mathfrak{A}$, представленных таким образом.

Лемма 1. Число систем размера n над $\mathfrak{A}$ равно

$$|\mathcal{S}_n| = \prod_{i=1}^{m} 2^{n^{k_i}}.$$

Доказательство. Прямой подсчёт. ■

Назовём алгебраическую систему $\mathfrak{A}$ *нетривиальной*, если существует система уравнений, которая не имеет решения над $\mathfrak{A}$. В противном случае $\mathfrak{A}$ *тривиальная*. Очевидно, что для тривиальных алгебраических систем проблема распознавания разрешимости систем уравнений разрешима за полиномиальное время.

Теорема 1. Проблема распознавания разрешимости систем уравнений над конечной нетривиальной алгебраической системой $\mathfrak{A}$ генерически разрешима за полиномиальное время.

Доказательство. Пусть S' — какая-то фиксированная система уравнений размера t, неразрешимая над $\mathfrak{A}$. Полиномиальный генерический алгоритм для распознавания разрешимости систем уравнений над $\mathfrak{A}$ работает на системе S размера n следующим образом:

1) Ищет в системе S подсистему, эквивалентную S': перебирает все выборки по t переменных из n переменных системы S; для каждой выборки ищет в S все предикаты из системы S' с учётом замены переменных S' соответствующими переменными из выборки. Число таких выборок $\mathrm{C}_n^t = O(n^t)$ полиномиально, и проверка каждой выборки делается за полиномиальное от n время.
2) Если эквивалентная подсистема нашлась, то выдаёт ответ «НЕТ».
3) Если нет, выдаёт ответ «НЕ ЗНАЮ».

Для доказательства генеричности этого алгоритма покажем, что множество систем уравнений, не содержащих подсистемы, эквивалентной S' (обозначим это множество A), является пренебрежимым. Рассмотрим множество систем B, в которых на переменных $\{x_1, \ldots, x_n\}$ запрещена подсистема S' для переменных $\{x_1, \ldots, x_t\}$, для переменных $\{x_{t+1}, \ldots, x_{2t}\}$, …, для переменных $\{x_{t([n/t]-1)+1}, \ldots, x_{t[n/t]}\}$. Здесь через $[x]$ обозначена целая часть числа x. Так как для систем из B запретов меньше, чем для систем из A, то $A \subseteq B$.

Обозначим через I множество индексов тех предикатов из P_i, $i = 1, \ldots, m$, сигнатуры σ, которые встречаются в системе уравнений S'. Можно подсчитать, что

$$|B_n| = \prod_{i \notin I} 2^{n^{k_i}} \prod_{i \in I} 2^{n^{k_i} - t^{k_i}[n/t]} (2^{t^{k_i}} - 1)^{[n/t]}.$$

Это следует из того, что в таблицах включения для каждого предиката с индексом из I для систем из множества B «запрещены» $[n/t]$ подтаблиц размера t, соответствующих предикатам из системы S'. Эти подтаблицы имеют по $2^{t^{k_i}}$ мест для расстановки нулей и единиц.

Теперь запишем:

$$\rho(B) = \lim_{n\to\infty} \frac{|B_n|}{|\mathcal{S}_n|} = \lim_{n\to\infty} \frac{\prod\limits_{i \notin I} 2^{n^{k_i}} \prod\limits_{i \in I} 2^{n^{k_i} - t^{k_i}[n/t]} (2^{t^{k_i}} - 1)^{[n/t]}}{\prod\limits_{i=1}^{m} 2^{n^{k_i}}} =$$

$$= \lim_{n\to\infty} \frac{\prod\limits_{i \in I} (2^{t^{k_i}} - 1)^{[n/t]}}{\prod\limits_{i \in I} 2^{t^{k_i}[n/t]}} = \prod_{i \in I} \lim_{n\to\infty} \left(1 - 2^{-t^{k_i}}\right)^{[n/t]} = 0.$$

Это доказывает, что множество B является пренебрежимым, а значит, его подмножество A тем более пренебрежимо. ■

3. Проблема поиска решения систем уравнений

Напомним, что проблема поиска решения систем уравнений над алгебраической системой $\mathfrak{A}$ состоит в том, что по заданной произвольной разрешимой над $\mathfrak{A}$ системе уравнений требуется найти любое её решение. Обозначим эту проблему $\mathcal{SEP}_{\mathfrak{A}}$. Для неё также не известно полиномиальных алгоритмов.

Рассмотрим бесконечную последовательность систем уравнений

$$\sigma = \{S_1, S_2, \ldots, S_n, \ldots\},$$

такую, что S_n имеет размер n для $n = 1, 2, 3, \ldots$ Для каждой последовательности систем σ определим подпроблему поиска решения систем уравнений $\mathcal{SEP}_{\mathfrak{A}}(\sigma)$ как ограничение исходной проблемы $\mathcal{SEP}_{\mathfrak{A}}$ на множество входов

$$\{S : S \cong S_n, S_n \in \sigma, n \in \mathbb{N}\}.$$

Здесь $S_1 \cong S_2$ означает, что системы S_1 и S_2 — это системы от одного множества переменных $\{x_1, \ldots, x_n\}$ и S_1 получена из S_2 некоторой перестановкой переменных.

Лемма 2. Если не существует полиномиального вероятностного алгоритма для решения проблемы $\mathcal{SEP}_{\mathfrak{A}}$, то найдётся последовательность систем σ, такая, что не существует полиномиального вероятностного алгоритма для решения проблемы $\mathcal{SEP}_{\mathfrak{A}}(\sigma)$.

Доказательство. Пусть $P_1, P_2, \ldots$ — все полиномиальные вероятностные алгоритмы. Если не существует полиномиального вероятностного алгоритма для проблемы $\mathcal{SEP}_{\mathfrak{A}}$, то для любого вероятностного полиномиального алгоритма P_n найдётся бесконечно много систем, для которых алгоритм P_n не может решить $\mathcal{SEP}_{\mathfrak{A}}$. Поэтому можно выбрать такую последовательность систем $\sigma' = \{S_1, S_2, \ldots, S_n, \ldots\}$, что алгоритм P_n не может решить $\mathcal{SEP}_{\mathfrak{A}}$ для S_n для всех n. Более того, можно считать, что σ' упорядочена по возрастанию размеров. Теперь можно расширить последовательность σ' до последовательности σ с системами S_n для всех размеров n. Из построения σ следует, что не существует полиномиального вероятностного алгоритма для решения проблемы $\mathcal{SEP}_{\mathfrak{A}}(\sigma)$. ■

Из определения видно, что множество всех входов размера n для проблемы $\mathcal{SEP}_{\mathfrak{A}}(\sigma)$ выглядит так:

$$I_n = \{S : S \cong S_n, S_n \in \sigma\}.$$

Лемма 3. Пусть σ — произвольная последовательность систем уравнений. Если существует генерический полиномиальный алгоритм, решающий проблему $\mathcal{SEP}_{\mathfrak{A}}(\sigma)$, то существует вероятностный полиномиальный алгоритм, решающий эту проблему на всём множестве входов.

Доказательство. Допустим, что существует генерический полиномиальный алгоритм $\mathcal{A}$, решающий проблему поиска решения систем уравнений $\mathcal{SEP}_{\mathfrak{A}}(\sigma)$. Построим вероятностный полиномиальный алгоритм $\mathcal{B}$, решающий эту проблему на всём множестве входов. На системе S размера n алгоритм $\mathcal{B}$ работает следующим образом:

1) Запускает алгоритм $\mathcal{A}$ на S.
2) Если $\mathcal{A}(S) \neq \square$, то $\mathcal{B}$ выдаёт ответ $\mathcal{A}(S)$ и останавливается, иначе идёт на шаг 3.
3) Генерирует случайно и равномерно перестановку π на множестве номеров переменных $\{x_1, \ldots, x_n\}$ и вычисляет систему $S' = \pi(S)$.

4) Запускает алгоритм $\mathcal{A}$ на S'.
5) Если $\mathcal{A}(S') = \square$, то выдаёт ответ $(a, \ldots, a)$, где $a \in A$, — возможно, неправильный.
6) Если $\mathcal{A}(S') = \{a_1, \ldots, a_n\}$ — решение системы S', то

$$\pi^{-1}(a_1, \ldots, a_n) = \left(a_{\pi^{-1}(1)}, \ldots, a_{\pi^{-1}(n)}\right)$$

является решением системы $S = \pi^{-1}(S')$.

Для доказательства корректности работы вероятностного алгоритма надо показать, что вероятность того, что $A(S') = \square$, меньше $1/3$. Заметим, что $\pi(S)$ при варьировании перестановки π пробегает всё множество входов размера n. Множество $\{S : A(S) = \square\}$ пренебрежимо, поэтому вероятность того, что $A(S') = \square$, стремится к нулю при увеличении n. ■

Теорема 2. Если для проблемы поиска решения систем уравнений над алгебраической системой $\mathfrak{A}$ не существует вероятностного полиномиального алгоритма, то существует последовательность систем уравнений σ, такая, что для решения проблемы $\mathcal{SEP}_{\mathfrak{A}}(\sigma)$ не существует генерического полиномиального алгоритма.

Доказательство. Пусть для проблемы $\mathcal{SEP}_{\mathfrak{A}}$ нет вероятностного полиномиального алгоритма. По лемме 2 найдётся такая последовательность систем σ, что и для $\mathcal{SEP}_{\mathfrak{A}}(\sigma)$ нет полиномиального вероятностного алгоритма. Теперь если допустить, что для $\mathcal{SEP}_{\mathfrak{A}}(\sigma)$ существует полиномиальный генерический алгоритм, то по лемме 3 для $\mathcal{SEP}_{\mathfrak{A}}(\sigma)$ найдётся полиномиальный вероятностный алгоритм. Полученное противоречие доказывает теорему. ■

ЛИТЕРАТУРА

1. *Baumslag G., Myasnikov A., and Remeslennikov V.* Algebraic geometry over groups I. Algebraic sets and ideal theory // J. Algebra. 1999. V. 219. No. 1. P. 16–79.
2. *Shevlyakov A. N.* Equationally Noetherian varieties of semigroups and B. Plotkin's problem // Сиб. электрон. матем. изв. 2023. Т. 20. № 2. С. 724–734.
3. *Шевляков А. Н.* Сплетения полугрупп и проблема Б. И. Плоткина // Алгебра и логика. 2023. Т. 62. № 5. С. 665–691.
4. *Shevlyakov A. N.* On disjunctions of algebraic sets in completely simple semigroups // Communications in Algebra. 2017. V. 45. No. 9. P. 3757–3767.
5. *Шевляков А. Н.* Об объединении решений систем уравнений в полугруппах с конечным идеалом // Алгебра и логика. 2016. Т. 55. № 1. С. 87–105.
6. *Рыбалов А. Н.* О сложности решения уравнений над графами // Сиб. электрон. матем. изв. 2024. Т. 21. № 1. С. 62–69.
7. *Nikitin A. and Shevlyakov A.* On radicals over strict partial order sets // J. Phys.: Conf. Ser. 2021. V. 1791. No. 012080. 6 p.
8. *Goldmann M. and Russell A.* The complexity of solving equations over finite groups // Information and Computation. 2002. V. 178. No. 1. P. 253–262.
9. *Klima O., Tesson P., and Therien D.* Dichotomies in the complexity of solving systems of equations over finite semigroups // Theory Comput. Syst. 2007. V. 40. P. 263–297.
10. *Ильев А. В., Ильев В. П.* Алгоритмы для решения систем уравнений над различными классами конечных графов // Прикладная дискретная математика. 2021. № 53. С. 89–102.
11. *Kapovich I., Miasnikov A., Schupp P., and Shpilrain V.* Generic-case complexity, decision problems in group theory and random walks // J. Algebra. 2003. V. 264. No. 2. P. 665–694.

12. *Rybalov A. and Shevlyakov A.* Generic complexity of solving of equations in finite groups, semigroups and fields // J. Phys.: Conf. Ser. 2021. V. 1901. No. 012047. 8 p.
13. *Impagliazzo R. and Wigderson A.* P = BPP unless E has subexponential circuits: Derandomizing the XOR Lemma. Proc. 29th STOC. El Paso: ACM, 1997. P. 220–229.
14. *Вялый М., Китаев А., Шень А.* Классические и квантовые вычисления. М.: МЦНМО, ЧеРо. 1999. 192 с.

REFERENCES

1. *Baumslag G., Myasnikov A., and Remeslennikov V.* Algebraic geometry over groups I. Algebraic sets and ideal theory. J. Algebra, 1999, vol. 219, no. 1, pp. 16–79.
2. *Shevlyakov A. N.* Equationally Noetherian varieties of semigroups and B. Plotkin's problem. Sib. Elektron. Mat. Izv., 2023, vol. 20, no. 2, pp. 724–734.
3. *Shevlyakov A. N.* Wreath products of semigroups and Plotkin's problem. Algebra and Logic, 2023, vol. 62, no. 5, pp. 448–467.
4. *Shevlyakov A. N.* On disjunctions of algebraic sets in completely simple semigroups. Communications in Algebra, 2017, vol. 45, no. 9, pp. 3757–3767.
5. *Shevlyakov A. N.* Combining solutions for systems equations in semigroups with finite ideal. Algebra and Logic, 2016, vol. 55, no. 1, pp. 58–71.
6. *Rybalov A. N.* O slozhnosti resheniya uravneniy nad grafami [On complexity of solving of equations over graphs]. Sib. Electr. Mat. Izv., 2024, vol. 21, no. 1, pp. 62–69. (in Russian)
7. *Nikitin A. and Shevlyakov A.* On radicals over strict partial order sets. J. Phys.: Conf. Ser., 2021, vol. 1791, no. 012080, 6 p.
8. *Goldmann M. and Russell A.* The complexity of solving equations over finite groups. Information and Computation, 2002, vol. 178, no. 1, pp. 253–262.
9. *Klima O., Tesson P., and Therien D.* Dichotomies in the complexity of solving systems of equations over finite semigroups. Theory Comput. Syst., 2007, vol. 40, pp. 263–297.
10. *Il'ev A. V. and Il'ev V. P.* Algoritmy dlya resheniya sistem uravneniy nad razlichnymi klassami konechnykh grafov [Algorithms for solving systems of equations over various classes of finite graphs]. Prikladnaya Diskretnaya Matematika, 2021, no. 53, pp. 89–102. (in Russian)
11. *Kapovich I., Miasnikov A., Schupp P., and Shpilrain V.* Generic-case complexity, decision problems in group theory and random walks. J. Algebra, 2003, vol. 264, no. 2, pp. 665–694.
12. *Rybalov A. and Shevlyakov A.* Generic complexity of solving of equations in finite groups, semigroups and fields. J. Phys.: Conf. Ser., 2021, vol. 1901, no. 012047, 8 p.
13. *Impagliazzo R. and Wigderson A.* P = BPP unless E has subexponential circuits: Derandomizing the XOR Lemma. Proc. 29th STOC, El Paso, ACM, 1997, pp. 220–229.
14. *Vyalyy M., Kitaev A., and Shen' A.* Klassicheskie i kvantovye vychisleniya [Classical and Quantum Computations]. Moscow, MCCME Publ., 1999. 192 p. (in Russian)

ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ В ДИСКРЕТНОЙ МАТЕМАТИКЕ

УДК 519.8 DOI 10.17223/20710410/70/7

КАЛЕНДАРНОЕ ПЛАНИРОВАНИЕ ИНВЕСТИЦИОННЫХ ПРОЕКТОВ ПРИ ВОЗМОЖНОСТИ КРЕДИТОВАНИЯ[1]

С. А. Малах, В. В. Сервах

Институт математики им. С.Л. Соболева СО РАН, г. Омск, Россия

E-mail: malahsveta@mail.ru, svv_usa@rambler.ru

Рассматривается задача календарного планирования проектов. Первая часть посвящена обзору различных постановок задачи в условиях ограничения на ресурсы. Во второй части исследуется вариант задачи с критерием максимизации чистой приведённой прибыли всего проекта. Основное внимание уделяется постановке, возникающей при реализации крупномасштабных проектов, когда ресурсы могут быть заменены их денежным эквивалентом. В этом случае в модели используется единственный вид ресурса — финансовый. Описана традиционная постановка задач и предлагается новый подход к их моделированию: вместо ограничений на ресурсы вводится оплата за их использование. Инструментарием оплаты является кредитование по фиксированной ставке. При таком подходе любое согласованное с частичным порядком расписание становится допустимым. Описана модель для расчёта потенциальных возможностей проекта, предложен алгоритм вычисления собственной прибыли при заданном расписании выполнения работ, исследуется вычислительная сложность задачи построения оптимального расписания, выделен полиномиально разрешимый случай задачи при возможности кредитования.

Ключевые слова: *календарное планирование, инвестиционные проекты, чистая приведённая прибыль, кредитование.*

INVESTMENT PROJECT SCHEDULING WITH LENDING

S. A. Malakh, V. V. Servakh

Sobolev Institute of Mathematics, Siberian Branch of the Russian Academy of Sciences, Omsk, Russia

This paper addresses the project scheduling problem. It provides an overview of various problem formulations under resource constraints, including those aimed at maximizing the Net Present Value of the entire project. Special attention is given to modeling scenarios typical of large-scale projects, where traditional resources can be substituted with their monetary equivalents. In such cases, the model is reduced to a single type of resource: financial resources. The standard problem formulation is described, and a novel modeling approach is proposed: instead of hard resource constraints, additional resource usage incurs a cost. This cost is modeled through

[1]Работа выполнена в рамках госзадания ИМ СО РАН, проект FWNF-2022-0020.

borrowing at a fixed interest rate. Under this framework, any schedule consistent with the partial order of activities becomes feasible. A recursive procedure is proposed for calculating net profit given a fixed activity schedule. It is shown that determining a schedule that maximizes net profit is strongly NP-hard. A special case of the problem is identified as polynomially solvable when the total number of profitable, technologically independent activities is bounded by a constant. A model is also presented to estimate the potential of a project under full self-financing. The question of whether this problem is polynomially solvable remains open. To address it, an approximate integer linear programming model with a unimodular matrix is proposed. However, the complexity status of this formulation likewise remains unresolved.

Keywords: *scheduling, investment project, Net Present Value, lending.*

Введение

Проектом будем называть множество технологически взаимосвязанных работ $i \in V$, где $V = \{1, 2, \ldots, n\}$ — множество всех работ проекта, выполнение которых направлено на достижение определённой цели. Взаимосвязь определяется частичным порядком E, а сам проект задаётся графом $G = (V, E)$. Для каждой работы $i \in V$ известна длительность её выполнения p_i. В выбранных единицах измерения времени величины p_i являются целочисленными. Задача минимизации общего срока выполнения проекта была успешно решена в 1958 г. при реализации проекта создания ракетной системы «Полярис» [1]. Проект, состоящий из 60 тыс. работ, удалось закончить на два года раньше ожидаемого срока. Примерно в это же время при планировании работ по модернизации заводов фирмы «Дюпон» был предложен метод критического пути [2]. Эти разработки получили широкое практическое применение благодаря простоте, наглядности и эффективности их использования. Они позволяют рассчитывать потребности в ресурсах на каждом этапе реализации проекта.

Трудности при составлении расписания выполнения работ возникли, когда проекты требовалось реализовывать в условиях ограничения на ресурсы. Такая задача получила название Resource Constrained Project Scheduling Problem (RCPSP) и заключается в том, чтобы планировать работы с учётом их приоритета и ограничений на ресурсы, при этом время выполнения должно быть минимальным. Математическая модель, представляющая RCPSP, была разработана в 1969 г. [3]. В 1983 г. доказана сильная NP-трудность этой задачи [4]. В дальнейшем были предложены разнообразные постановки задачи планирования проектов с ограниченными ресурсами, в том числе и с различными критериями.

Задачу с критерием чистой приведенной прибыли проекта впервые рассмотрел A. H. Russell [5] в 1970 г. В работе [6] для планирования крупномасштабных проектов предложено свести задачу к ограничениям на ресурс финансового типа, а другие ресурсы трансформировать в их денежный эквивалент. В [7] доказана сильная NP-трудность этой задачи. Задача с кредитованием была представлена в [8], а её сильная NP-трудность доказана в [9]. В настоящей работе мы развиваем подходы к решению задачи с кредитованием.

В п. 1 сделан обзор различных постановок задачи, описана общая концепция ресурсных ограничений. Пункт 2 посвящён классической постановке задачи с критерием максимизации чистой приведённой прибыли. В п. 3 описана постановка задачи максимизации собственной прибыли с учётом возможности кредитования проекта. В п. 4 предложен подход к моделированию задачи, в котором ресурсные ограничения отсутствуют, а недостающие ресурсы покрываются за счёт кредитов. Описан алгоритм

расчёта собственной прибыли для заданного расписания выполнения работ. В п. 5 проведён анализ сложности предложенной постановки, выделен полиномиально разрешимый подслучай задачи. В п. 6 рассматривается задача оценки потенциальных возможностей проекта и предлагается подход к её решению.

1. Задача календарного планирования

Приведём классическую постановку задачи [10]. Рассмотрим два типа ресурсов: возобновимые и складируемые. К первым относится оборудование, рабочие, специалисты, производственные помещения, а к складируемым — расходные материалы, сырьё, финансы и т. д. Пусть W^r и W^a — множества видов возобновимых и складируемых ресурсов соответственно. Объём ресурса вида $w \in W^r \cup W^a$ на интервале $[t-1, t)$ обозначим $K^w(t)$, $t \in \mathbb{N}$. Возобновимые ресурсы доступны в течение всего интервала, а складируемые — на начало указанного периода. Без ограничения общности предполагается, что длительности всех работ целочисленные. Работа $i \in V$ на интервале $[\tau - 1, \tau)$, $\tau = 1, 2, \ldots, p_i$, потребляет $k_i^w(\tau)$ единиц ресурса вида w. Необходимо найти s_i — время начала выполнения работы $i \in V$. Обозначим через $N_t = \{i \in V : s_i < t \leqslant s_i + p_i\}$ множество работ, выполняемых на интервале $[t-1, t)$. Расписание $(s_1, s_2, \ldots, s_n)$ называется допустимым, если:

— соблюдается технологическая последовательность выполнения работ:

$$s_i + p_i \leqslant s_j, \quad (i, j) \in E;$$

— соблюдаются ограничения на ресурсы складируемого типа:

$$\sum_{\tau=1}^{t} \sum_{i \in N_\tau} k_i^w(\tau - s_i) \leqslant \sum_{\tau=1}^{t} K^w(\tau), \quad w \in W^a, \ t = 1, 2, \ldots;$$

— соблюдаются ограничения на ресурсы возобновимого типа:

$$\sum_{i \in N_\tau} k_i^w(\tau - s_i) \leqslant K^w(t), \quad w \in W^r, \quad t = 1, 2, \ldots$$

Если имеются только ресурсы складируемого типа, то задача минимизации общего времени завершения всех работ полиномиально разрешима [10]. Для других критериев задача NP-трудна в сильном смысле. Если ресурсы возобновимые, то даже поиск допустимого решения является сильно NP-трудной задачей. В этом случае легко сконструировать примеры, когда хотя бы одно допустимое решение есть, но отыскать его трудно. Для этого достаточно взять произвольное расписание выполнения работ и рассчитать, сколько в точности требуется ресурсов для его реализации. Этот минимально необходимый уровень ресурсов зафиксировать как входные данные. Далее информацию о расписании убрать. В результате получаем пример с непустым множеством допустимых решений. Найти допустимое расписание такого примера — очень трудная задача, так как ограничения на ресурсы получаются очень жёсткие. Такие примеры являются наиболее сложными в дискретной оптимизации.

В литературе представлено множество идей по поводу того, каким образом можно ослабить жёсткость ресурсных ограничений. В [11, 12] возобновимые ресурсы заменяются складируемыми, что позволяет получить нижние оценки оптимального решения задачи. Хорошо известна задача trade-off, в которой длительность работы зависит от выделенных на неё ресурсов [13]: имеется возможность перераспределить ограниченные ресурсы в пользу критических работ. С другой стороны, в случае нехватки ресурсов в некоторый момент времени можно увеличить длительность работ и уложиться

в ограничения. Ещё одним важным подходом, направленным на более гибкую работу с ресурсными ограничениями, является приобретение ресурсов. Задача минимизации затрат на закупку ресурсов впервые была рассмотрена в 1984 г. [14]. Из последних исследований отметим работы [15, 16]. В [17, 18], помимо составления расписания, решается задача, в которой необходимо определить дополнительные параметры: в какие моменты времени, где и в каком количестве заказывать материалы для проекта. В работах [15, 18] учитывается ещё и скидка на количество заказов. Недостатком такого подхода является необходимость оптимизации второй целевой функции — стоимости закупленных ресурсов.

В настоящее время представления о ресурсах существенно расширились. Прежде всего отметим понятие регенерации ресурсов [19, 20]. Предположим, что каждая выполняемая работа потребляет некоторое количество единиц ресурса, это интерпретируется как расход, и воспроизводит другое количество единиц ресурса после своего завершения, которое интерпретируется как доход. Для возобновимых ресурсов расход равен доходу, а для складируемых доход равен нулю. Регенерация ресурсов предполагает наличие коэффициента β восстановления ресурсов после выполнения работы. Ранее рассматривалось два значения этого коэффициента: если $\beta = 1$, то ресурс является возобновимым, если $\beta = 0$ — складируемый. В более общем случае значение β может принимать любые значения. В качестве примера можно рассмотреть работы [21, 22], в которых исследуется модель промежуточного переселения жителей в рамках проекта перепланировки районов города. Описанная задача отличается от классической задачи календарного планирования с ограниченными ресурсами тем, что доход любой выполненной работы может быть не только положительным или равным нулю, но и отрицательным. Авторы работы [23] рассматривают особый случай с парами работ, когда первая работа занимает ресурс в момент своего начала, при этом такое же количество мощности высвобождается по завершении второй работы. Связанные ресурсы называются ресурсами «брать — давать».

Этот подход можно обобщить, предполагая воспроизведение ресурса не только после окончания работы, но и в процессе её выполнения, а также с временным лагом после её завершения. Были введены понятия частично возобновимых ресурсов, а также понятие последовательно восстановимых ресурсов. В [24] авторы используют частично возобновимые ресурсы в рамках задачи планирования нескольких проектов. В [25] рассматривается задача с частично возобновимыми ресурсами и минимальными и максимальными временными задержками.

Отметим ещё несколько аспектов, связанных с ресурсами [26]. Это понятие общего или совокупного ресурса, когда подмножество работ использует общий ресурс. Другое направление связано с ресурсами, которые обладают множественными навыками (multiple skills), а каждая работа требует наличия определённого набора этих навыков. Некоторые исследователи рассматривают проблему множественных навыков с эффектами обучения или усталости. Но исследование ресурсов таких типов выходит за рамки данной работы.

Хотя RCPSP является классической моделью, она не может охватить все ситуации, возникающие на практике. Поэтому многие исследователи разработали более общие задачи планирования проектов, часто используя стандартную RCPSP в качестве отправной точки. В [27] собрано более 60 статей, охватывающих многие важные модели и методы планирования проектов. S. Hartmann и D. Briskorn [26] представили широкий обзор вариантов и расширений RCPSP, которые были предложены другими авторами.

2. Критерии, основанные на чистой приведённой стоимости

Одним из важных вариантов рассматриваемой задачи является планирование инвестиционных проектов, основная цель которых направлена на получение прибыли от выполнения комплекса технологически взаимосвязанных работ. В такой постановке возникают денежные потоки, а для их сравнения в различные моменты времени используется операция дисконтирования. При совершении финансовых операций предполагается, что имеется возможность альтернативного безрискового ликвидного размещения капитала под ставку r_0 за единичный период времени. Тогда капитал K_0, которым располагает инвестор в момент t_0, к моменту t увеличивается до величины $K_t = K_0(1 + r_0)^{t-t_0}$. Тем самым капитал K_t в момент времени t эквивалентен капиталу $K_t/(1 + r_0)^{t-t_0}$ в момент t_0. Операция приведения к начальному моменту времени называется дисконтированием и позволяет сравнивать деньги в разные моменты времени.

В литературе рассматриваются разные подходы к моделированию инвестиционных проектов. Отток денежных средств вызван выполнением работ и использованием ресурсов, приток денежных средств происходит по завершении определённых частей проекта. Это приводит к необходимости максимизировать чистую текущую стоимость (Net Present Value, NPV) проекта. В дополнение к стандартному приоритету и ограничениям по ресурсам учитывается ограничение по срокам. RCPSP с целью максимизации NPV изучалось в [28–31]. Эти исследования основаны на непрерывном начислении сложных процентов, то есть денежные потоки дисконтируются с коэффициентом $e^{-\beta t}$. В [32] рассматривается та же ситуация, но используется начисление сложных процентов за период с коэффициентом дисконтирования $(1 + \alpha)^{-t}$. Однако эти два типа дисконтирования существенно не отличаются, поскольку могут быть конвертированы друг в друга. В работе [33] расширяется RCPSP с целевой функцией NPV — рассматривается отток денежных средств либо в начале, либо в конце работы, либо на протяжении всего периода её выполнения (приток денежных средств происходит только в конце периода). Ограничение гарантирует, что капитал никогда не станет отрицательным.

В [34] применяется аналогичный подход, где три варианта поступления платежей применяются также к притоку денежных средств. В [35] рассматриваются платежи в регулярные и нерегулярные моменты времени, а также платежи, связанные с работами, которые включены как в одномодовую, так и в многорежимную RCPSP.

В [36] предлагается рассматривать цель, основанную на чистой приведённой стоимости, которая учитывает приток денежных средств после завершения деятельности, затраты на ресурсные мощности, а также бонусы и штрафные платежи в зависимости от завершения проекта в отношении срока выполнения. Учитывается также уровень инфляции.

Затраты на объём ресурсов относятся только к интервалу времени, в течение которого ресурс фактически используется. В [37] исследуется RCPSP с целью максимизировать чистую текущую стоимость и дополнительный единый непрерывный ресурс.

В [38] максимизируется чистая приведённая стоимость; в [39] в RCPSP с заданными сроками выполнения работ минимизируется чистая приведённая стоимость штрафов за нарушение директивных сроков этих работ.

В данной работе предлагается подход, который может быть использован для моделирования крупномасштабных проектов и включает в себя большинство описанных выше понятий. Кроме того, в модели удаётся избежать многокритериальности.

Заметим, что в крупномасштабных проектах можно заменить все ресурсы их денежным эквивалентом и рассматривать только один вид ресурса — финансовый. Имея финансовый ресурс, рабочих можно нанять, необходимые помещения и оборудование арендовать, взять в лизинг или купить, возможно, с последующей продажей. Если же финансового ресурса не хватает, то его можно приобрести с помощью кредита. В итоге все платежи, связанные с работой $i \in V$, сводятся к одному потоку $(c_i(0), c_i(1), \ldots, c_i(p_i))$, где $c_i(\tau)$ — баланс платежей работы i в момент времени $\tau = 0, 1, \ldots, p_i$. Под балансом платежей будем понимать разность между поступлениями и расходами. Если $c_i(\tau) < 0$ — затраты превосходят поступления, если $c_i(\tau) > 0$ — поступления больше затрат. Величина

$$NPV_i = \sum_{\tau=0}^{p_i} \frac{c_i(\tau)}{(1 + r_0)^\tau}$$

называется чистой прибылью работы $i \in V$, *приведённой к началу её выполнения*. Предполагается, что каждая работа выполняется без прерываний. Наличие ресурсов в момент времени t также задаём в денежном эквиваленте совокупной величиной $K(t)$, $t = 0, \ldots, T$, где T — горизонт планирования проекта.

Обозначим, как и ранее, через s_i момент начала выполнения работы $i \in V$. Вектор $\mathbf{S} = (s_1, s_2, \ldots, s_n)$ задаёт расписание выполнения работ проекта. Так как p_i — целые числа и потоки платежей дискретны, достаточно рассмотреть расписания с целыми значениями s_i.

В данной постановке платежи привязаны к моментам $\tau = 0, 1, \ldots, p_i$. Поэтому переопределим множество N_t как множество работ, выполняемых в момент $t \in \in \{0, 1, \ldots, T\}$, то есть $N_t = \{i \in V : s_i \leqslant t \leqslant s_i + p_i\}$. Расписание $\mathbf{S}$ называется допустимым, если:

— проект завершается к моменту T:

$$s_i + p_i \leqslant T, \quad i \in V;$$

— сохраняется заданный частичный порядок выполнения работ:

$$s_i + p_i \leqslant s_j, \quad (i, j) \in E;$$

— в каждый момент времени с учётом реинвестирования дохода и размещения свободного капитала под ставку r_0 финансовых ресурсов достаточно для выполнения работ проекта:

$$\sum_{t=0}^{t^*} \frac{K(t)}{(1 + r_0)^t} + \sum_{t=0}^{t^*} \sum_{i \in N_t} \frac{c_i(t - s_i)}{(1 + r_0)^t} \geqslant 0, \quad t^* = 0, \ldots, T.$$

Требуется определить допустимое расписание выполнения работ, при котором чистая приведённая прибыль всего проекта будет максимальной. Чтобы просуммировать прибыль от всех работ, требуется величины NPV_j привести к моменту времени $t = 0$. Тогда целевая функция будет выглядеть следующим образом:

$$NPV_{rc}(\mathbf{S}) = \sum_{i \in V} \frac{NPV_i}{(1 + r_0)^{s_i}} \to \max_{\mathbf{S}}.$$

Данная задача является NP-трудной в сильном смысле [7]. Различные варианты этой модели исследовались в [28, 29, 33, 35].

Отметим, что критерий чистой приведённой прибыли удобно использовать для достижения различных целей проекта, в частности для минимизации общего времени его выполнения. Достаточно ввести фиктивную заключительную работу, которая зависит от всех работ проекта, и сделать её доход больше, чем максимально возможный общий доход проекта. Тогда при оптимизации NPV-критерия эта работа автоматически будет выполняться в ранние сроки, и весь проект завершится как можно раньше.

3. Задача планирования инвестиционных проектов при возможности кредитования

Обобщим модель, допустив возможность привлечения за определённую плату дополнительных ресурсов. Рассмотрим проблемы финансирования проекта. Основными источниками финансирования инвестиционного проекта являются собственные средства, кредиты и средства соинвесторов. В описанной выше постановке финансирование проекта полностью осуществляется за счёт собственных средств инвестора. Если их не хватает, то либо берётся кредит, либо привлекаются соинвесторы. Соинвестирование и кредитование проекта — это два разных подхода к финансированию, каждый из которых имеет свои особенности и цели.

Соинвестирование подразумевает, что каждый соинвестор вносит свою долю капитала и, как правило, получает пропорциональную долю в прибыли и убытках проекта. Соинвесторы могут участвовать в управлении проектом, принимая решения о его развитии, что может привести к более активному вовлечению в процесс.

Кредитование проекта — это процесс получения заёмных средств от финансовых учреждений под определённые условия (процентная ставка, график погашения и т. д.). Заёмщик обязан возвращать заём с процентами независимо от успеха или неудачи проекта. Кредиторы, как правило, не участвуют в управлении проектом, но могут устанавливать условия и требования к заёмщику.

Каждый из этих методов имеет свои преимущества и недостатки, и выбор между ними зависит от конкретных обстоятельств проекта и целей инвесторов. Далее рассмотрим только вариант с кредитованием проекта, а соинвестирование оставим для дальнейших исследований.

При моделировании кредитов возникает большое количество дополнительных характеристик, таких, как процентная ставка, тип кредита, его размер, срок, схема выплат и т. д. Сделаем некоторое предположение, которое упрощает постановку, но по существу не влияет на адекватность модели. Будем считать, что: 1) в любой момент времени можно взять кредит; 2) процентная ставка фиксирована и не меняется в зависимости от срока и суммы кредита; 3) любая сумма кредита доступна в любое время; 4) погашение кредита допускается в любое время. Тогда любой кредит можно представить в виде потока платежей

$$\begin{pmatrix} t_0 & t_1 & \dots & t_k \\ \xi_0 & \xi_1 & \dots & \xi_k \end{pmatrix},$$

обладающего свойством

$$\sum_{i=0}^{k} \frac{\xi_i}{(1+r)^{t_i - t_0}} = 0,$$

где r — процентная ставка по кредиту за единичный период времени.

Кредит любой сложности можно разбить на последовательность простых кредитов, реализуемых по схеме «взял — вернул с процентами». Так как у нас время дискретно и t принимает целочисленные значения, то справедливо следующее

Утверждение 1 [9]. При фиксированной процентной ставке кредит любого вида можно разбить на эквивалентную последовательность кредитов единичной длительности.

Действительно, при дискретном времени в очередной целочисленный момент времени возвращается весь долг по кредиту вместе с начисленными процентами. Кредит полностью закрывается. И в тот же момент открываем новый кредит на ту же сумму. Полученная последовательность потоков платежей соответствует исходной схеме кредита.

Такой подход позволяет ввести только один дополнительный тип переменных $D(t)$ — размер кредита, взятого в год t. Тогда модель с кредитами и реинвестированием дохода имеет следующий вид: построить расписание выполнения работ $\mathbf{S} = (s_1, s_2, \ldots, s_n)$, $s_i \in \{0, 1, \ldots, T - p_i\}$, при котором:

— соблюдается технологический порядок выполнения работ:

$$s_i + p_i \leqslant s_j, \quad (i, j) \in E;$$

— в каждый целочисленный момент времени $t^* \in \mathbb{Z}^+$ сохраняется неотрицательный платёжный баланс с учётом взятых кредитов, выплат по ним, реинвестирования дохода и размещения свободного капитала под ставку r_0:

$$\sum_{t=0}^{t^*} \left(\frac{K(t)}{(1+r_0)^t} + \sum_{i \in N_t} \frac{c_i(t - s_i)}{(1+r_0)^t} + \frac{D(t) - (1+r)D(t-1)}{(1+r_0)^t} \right) \geqslant 0;$$

— чистая приведённая прибыль с учётом выплат по кредитам достигает максимального значения:

$$NPV_{\text{loan}}(\mathbf{S}, \mathbf{D}) = \sum_{i \in V} \frac{NPV_i}{(1+r_0)^{s_i}} + \sum_{t=0}^{T} \frac{D(t) - (1+r)D(t-1)}{(1+r_0)^t} \to \max_{\mathbf{S},\mathbf{D}},$$

где T — горизонт планирования проекта; $D(-1) = 0$ и $D(T) = 0$.

Использование кредитов расширяет множество допустимых решений задачи, поэтому оптимальное решение NPV^*_{loan} должно быть по крайней мере не хуже, чем оптимальное решение NPV^*_{rc} для задачи без кредитов. Таким образом, если задача с критерием NPV_{rc} разрешима, то справедливо следующее неравенство:

$$NPV^*_{rc} \leqslant NPV^*_{\text{loan}}.$$

Выделим особенности построенной модели:

- цель — получение инвестором максимальной прибыли;
- используется единственный ресурс — финансовый;
- учитывается временной фактор стоимости денежных средств;
- имеется возможность использования кредита;
- полученный в процессе выполнения проекта доход реинвестируется.

Данная задача также является NP-трудной в сильном смысле [9]. Более того, в ней увеличилось количество переменных. Далее предлагается использовать другой подход к моделированию этой задачи, основные идеи которого изложены в [40].

4. Рекурсивный подход при планировании инвестиционных проектов

Пусть, как и прежде, r_0 — ставка альтернативного безрискового ликвидного размещения свободного капитала, r — ставка по кредиту. Для финансирования проекта в момент t имеется капитал K_t, где $t = 0, 1, \ldots, T$. Если в какой-то момент времени его не хватает, то инвестор берёт кредит. Заметим, что при заданном расписании выполнения работ кредитные заимствования определяются однозначно. Для этого в каждый целочисленный момент времени недостаток финансовых средств покрывается минимально необходимым кредитом, который, в соответствии с утверждением 1, возвращается в следующий целочисленный момент времени. В такой ситуации при заданном расписании выполнения работ собственную чистую приведённую прибыль проекта $NPV_{\text{taut}}(\mathbf{S})$ можно вычислять алгоритмически (алгоритм 1).

Алгоритм 1. Вычисление собственной чистой приведённой прибыли для заданного расписания

1: Сформируем общий поток платежей проекта для расписания $\mathbf{S}$. Для этого достаточно сложить потоки работ $(c_i(0), c_i(1), \ldots, c_i(p_i))$, привязывая их к моменту начала выполнения работы s_i. В результате получим общий поток платежей по проекту $(C_0, C_1, \ldots, C_t, \ldots, C_T)$ для данного расписания $\mathbf{S}$, где $T = \max\limits_{i \in V}(s_i + p_i)$. Данный поток платежей и поступлений не зависит от того, как этот проект будет финансироваться.

2: Обозначим через F_t текущий платёжный баланс на момент t с учётом использования кредитов и выплат по ним, $t = 0, 1, \ldots, T$. Первоначально полагаем $F_0 = K_0$.

3: **Если** $K_0 + C_0 < 0$, **то**
необходимо взять кредит по ставке r и $F_1 = (K_0 + C_0)(1 + r)$.

4: **Если** $K_0 + C_0 \geqslant 0$, **то**
свободные деньги размещаем под ставку r_0 и $F_1 = (K_0 + C_0)(1 + r_0)$.

5: **Для всех** $t = 1, 2, \ldots, T - 1$
значения F_{t+1} для $t = 1, 2, \ldots, T - 1$ вычисляются рекурсивно:
$F_{t+1} = (F_t + K_t + C_t)(1 + r)$, если $F_t + K_t + C_t < 0$;
$F_{t+1} = (F_t + K_t + C_t)(1 + r_0)$, если $F_t + K_t + C_t \geqslant 0$.

6: Дисконтируя величину $F_T + C_T$ к начальному моменту времени и вычитая вложенный капитал, получаем собственную чистую прибыль, приведённую к начальному моменту времени:

$$NPV_{\text{taut}}(\mathbf{S}) = \frac{F_T + C_T}{(1 + r_0)^T} - \sum_{i=0}^{T-1} \frac{K_t}{(1 + r_0)^t}.$$

7: **Вывести** $NPV_{\text{taut}}(\mathbf{S})$.

В конечном итоге требуется найти расписание выполнения работ проекта, при котором собственная прибыль будет наибольшей:

$$\begin{cases} NPV_{\text{taut}}(\mathbf{S}) \to \max\limits_{\mathbf{S}}, & \\ s_i + p_i \leqslant s_j, & (i, j) \in E, \\ s_i \in \{0, 1, \ldots, T - p_i\}, & i \in V. \end{cases}$$

Утверждение 2. Оптимальные значения целевых функций NPV^*_{taut} и NPV^*_{loan} совпадают.

Доказательство. Построенная модель отличается от модели с критерием NPV_{loan} дополнительным требованием использования минимально необходимого объёма кредита в любой момент времени. Понятно, что в оптимальном решении модели с ограничением значения D_t также будут минимально необходимыми. Оба подхода являются решением одной и той же задачи. ■

Трудоёмкость шага 1 алгоритма 1 составляет $\sum\limits_{i \in V} p_i$ операций. Шаг 3 реализуется за $O(T)$ операций. Учитывая, что длина входа задачи зависит линейно от $\sum\limits_{i \in V} p_i$ и T, алгоритм расчёта собственной прибыли для заданного расписания выполнения работ является полиномиальным.

5. Вычислительная сложность и подходы к решению задачи

Приведём несколько результатов по вычислительной сложности решения поставленной задачи.

Утверждение 3. Задача максимизации $NPV_{\text{taut}}(\mathbf{S})$ NP-трудна в сильном смысле.

Доказательство. В силу утверждения 2 при нахождении оптимума задачи с критерием NPV_{taut} мы находим и оптимум задачи с критерием NPV_{loan}. Так как последняя задача NP-трудна в сильном смысле [9], такую же сложность имеет исходная задача. ■

Заметим, что вклад работы i в целевую функцию составляет $\dfrac{NPV_i}{(1+r_0)^{s_i}}$. Если $NPV_i < 0$, то данная функция убывает с ростом s_i Следовательно, при увеличении момента s_i общая прибыль проекта возрастает. Это свойство позволяет выделить полиномиально разрешимый случай задачи. Если $NPV_i > 0$, то работу i будем называть прибыльной.

Теорема 1. Если число прибыльных работ ограничено константой m, то задача максимизации собственной прибыли $NPV_{\text{taut}}(\mathbf{S})$ полиномиально разрешима.

Доказательство. Доказательство основано на приведённом выше замечании. В оптимальном расписании работы, имеющие отрицательную чистую приведённую прибыль, выполняются как можно позднее. Таким образом, если зафиксировать моменты начала выполнения прибыльных работ S_i, то для всех остальных работ достаточно найти поздние моменты их начала. А так как $S_i \in \{0, 1, \ldots, T - p_i\}$, где T — горизонт планирования проекта, то трудоёмкость перебора расписаний составит не более $O(T^m)$ шагов. Для расчёта прибыли для каждого расписания требуется не более $\sum\limits_{i \in V} p_i$ операций. Следовательно, в данном случае алгоритм является полиномиальным. ■

Трудоёмкость $O(T^m)$ является грубой верхней оценкой и может быть уменьшена за счёт использования алгоритма из [6], основанного на схеме динамического программирования. Его трудоёмкость экспоненциально зависит от максимального числа технологически независимых прибыльных работ. Если число таких работ ограничено константой, то алгоритм становится полиномиальным. Это усиливает результат теоремы 1, но его доказательство выходит за рамки данной работы.

Плюсом построенной модели является то, что в ней отсутствуют ограничения на ресурсы, а значит, любое согласованное с частичным порядком расписание является допустимым. Это позволяет использовать широкий спектр метаэвристик, включая

эволюционные алгоритмы. Разработка таких алгоритмов является целью наших дальнейших исследований.

6. Оценка эффективности проекта

Рассмотрим задачу максимизации NPV в предположении, что ресурсы не ограничены. Величина

$$NPV(\mathbf{S}) = \sum_{j \in V} \frac{NPV_j}{(1+r_0)^{s_j}}$$

характеризует чистую приведённую прибыль проекта при отсутствии ограничений на ресурсы. Заметим, что для любого технологически допустимого расписания $\mathbf{S}$ выполняется неравенство

$$NPV_{\text{loan}}(\mathbf{S}) \leqslant NPV(\mathbf{S}).$$

Нахождение расписания, для которого $NPV(\mathbf{S})$ принимает наибольшее значение, позволяет:

1) определить потенциальные возможности проекта;
2) построить верхнюю оценку оптимума для задачи с ресурсными ограничениями;
3) найти оптимальное решение задачи в случае, когда собственных средств хватает на полное финансирование проекта.

Отметим, что для части работ $NPV_i < 0$. Это в основном стартовые работы. Но отрицательность NPV_i возможна также и для заключительных работ. Это происходит, когда в проект включены, например, социально значимые работы (благоустройство территории, строительство спортивной площадки, культурных объектов и т. д.) или работы, связанные с обеспечением безопасности объекта. Некоторые из них могут быть весьма затратными. Максимизация прибыли в этом случае приводит к тому, что для этих работ $s_i \to \infty$, то есть фактически работа $i \in V$ не будет выполнена. Чтобы избежать таких ситуаций, необходимо ввести либо директивный срок, либо горизонт планирования проекта, к которому все работы должны быть завершены, обозначим его через T. Таким образом, возникает следующая модель:

$$\begin{cases} NPV(\mathbf{S}) = \sum\limits_{i \in V} \dfrac{NPV_i}{(1+r_0)^{s_i}} \to \max\limits_{S}, & \\ s_i + p_i \leqslant s_j, & (i,j) \in E, \\ s_i \in \{0, 1, \ldots, T - p_i\}, & i \in V. \end{cases}$$

Данная задача представляет определённый математический интерес, но к настоящему моменту построить полиномиальный алгоритм её решения не удалось и вопрос о её вычислительной сложности остаётся открытым. Далее предлагается алгоритм построения приближённого решения, основанный на линеаризации целевой функции.

Целевая функция является линейной комбинацией выпуклых и вогнутых строго монотонных функций. При $NPV_j < 0$ функция $\dfrac{NPV_j}{(1+r_0)^{s_j}}$ возрастает, и значит, увеличение s_j приводит к увеличению прибыли. В результате эту работу необходимо выполнять как можно позднее. Если $NPV_j > 0$, то, наоборот, работа должна быть выполнена как можно раньше. Ограничением является наличие частичного порядка выполнения работ E.

Рассмотрим релаксационную модель задачи, в которой условие целочисленности отсутствует, то есть $0 \leqslant s_i \leqslant T - p_i$, $i \in V$. Компоненты градиента целевой функции знакопостоянны и не могут быть равны нулю. Значит, оптимум достигается на

границе. Более того, на грани меньшей размерности все свойства целевой функции сохраняются. Для граней $s_i = 0$ и $s_i = T - p_i$ это очевидно. На грани $s_i + p_i = s_j$ сумма двух членов равна

$$\frac{NPV_i}{(1+r_0)^{s_i}} + \frac{NPV_j}{(1+r_0)^{s_j}} = \frac{NPV_i}{(1+r_0)^{s_i}} + \frac{NPV_j}{(1+r_0)^{s_i+p_i}} = \frac{1}{(1+r_0)^{s_i}} \left(NPV_i + \frac{NPV_j}{(1+r_0)^{p_i}} \right) = \frac{NPV_{ij}}{(1+r_0)^{s_i}},$$

где NPV_{ij} — общая доходность работ i и j. Получаем такую же функцию со знакопостоянным градиентом. Таким образом, можно сделать вывод, что оптимальное решение релаксационной задачи достигается в вершине многогранника. А так как матрица ограничений унимодулярна, то оптимальное решение будет целочисленным.

К сожалению, это мало что даёт для поиска оптимального решения задачи. Опишем алгоритм, использующий линеаризацию целевой функции. Аппроксимация экспоненты $(1 + r_0)^{-s_i}$ линейной функцией даёт очень грубое приближение, иллюстрация при $r_0 = 0{,}1$ и $s_i \in \{0, \ldots, 40\}$ приведена на рис. 1.

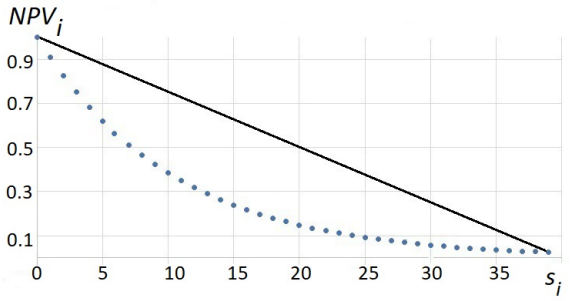

Рис. 1. Линейная аппроксимация NPV_i на интервале планирования 40 лет

Однако в нашей задаче есть особенности, которые можно использовать при аппроксимации. Реальные инвестиционные проекты на 40 лет не планируют. Если $s_i \in \{0, \ldots, 10\}$, то приближение становится значительно лучше (рис. 2). Если же процентная ставка повышается, то в силу экономических особенностей горизонт планирования проектов становится меньше и качество приближения не ухудшается.

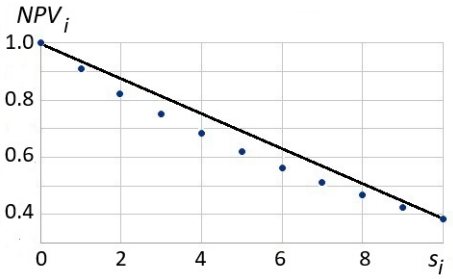

Рис. 2. Линейная аппроксимация NPV_i на интервале планирования 10 лет

Если меняется единица измерения, то соответственно меняется и процентная ставка. На рис. 3 рассматриваемая функция и её линейная регрессия отображены на интервале планирования 120 месяцев.

Следующий этап учитывает то обстоятельство, что переменная s_i изменяется в существенно меньшем диапазоне, чем $\{0, \ldots, T - p_i\}$. Опишем алгоритм 2 локализации значений переменной s_i.

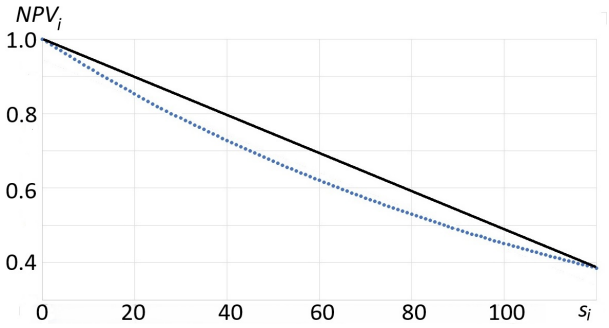

Рис. 3. Линейная аппроксимация NPV_i на интервале планирования 120 месяцев

Алгоритм 2. Локализация значений переменных s_i

1: **Для всех** $i \in V$
   находим ранние t_i^r и поздние t_i^p времена начала выполнения работы i.
2: **Для всех** $i \in V$, таких, что $NPV_i > 0$
3: работу i начинаем в срок t_i^r, а остальные работы выполняем как можно позднее. Этот срок берём за новое раннее время выполнения работ с неположительным значением NPV_i.
4: **Для всех** $i \in V$, таких, что $NPV_i < 0$
5: работу i начинаем в срок t_i^p, а остальные работы выполняем как можно раньше. Этот срок берём за новое позднее время выполнения работ с неотрицательным значением NPV_i.
6: **Для всех** $i \in V$
7: имеем $s_i \in [t_i^r, t_i^p]$

Функцию $(1+r_0)^{-s_i}$ на интервале $[t_i^r, t_i^p]$ аппроксимируем линейной функцией $a_i s_i + b_i$. Возможны два способа линеаризации. В первом строим линейную регрессию. Во втором проводим прямую через две крайние точки. Второй способ учитывает то обстоятельство, что функция строго монотонна и при прочих равных условиях оптимум достигается в крайнем положении.

Рассмотрим функцию $y_i = \dfrac{NPV_i}{(1+r_0)^{s_i}}$ на временном отрезке $[t_i^r, t_i^p]$. Обозначим $\alpha_1 = \dfrac{NPV_i}{(1+r_0)^{t_i^r}}$, $\alpha_2 = \dfrac{NPV_i}{(1+r_0)^{t_i^p}}$. Выпишем уравнение прямой в координатах (s_i, y_i), проходящей через точки $A(t_i^r, \alpha_1)$ и $B(t_i^p, \alpha_2)$:

$$\frac{t - t_i^r}{t_i^p - t_i^r} = \frac{y_i - \alpha_1}{\alpha_2 - \alpha_1}.$$

Уравнение прямой можно записать в виде

$$y_i = \frac{\alpha_2 - \alpha_1}{t_i^p - t_i^r} s_i - \frac{t_i^r(\alpha_2 - \alpha_1) - (t_i^p - t_i^r)\alpha_1}{t_i^p - t_i^r} = a_i s_i - b_i.$$

Поскольку константа b_i в целевой функции не влияет на решение, получаем следующую задачу линейного программирования:

$$\begin{cases} \sum\limits_{i \in V} a_i s_i \to \max\limits_{\mathbf{S}}, \\ s_i + p_i \leqslant s_j, \qquad (i,j) \in E, \\ t_i^r \leqslant s_i \leqslant t_i^p, \qquad i \in V. \end{cases}$$

Матрица данной задачи унимодулярна и параметры p_i, t_i^r, t_i^p целочисленные. Значит, оптимальное решение тоже будет целочисленным. Остаётся подставить полученное решение в целевую функцию.

Проведен эксперимент на задачах с 30 работами. Проекты формировались на основе задач из библиотеки PSLIB. Потоки платежей работ генерировались случайным образом. Точное решение задачи было получено с помощью решателя NLP в среде GAMS. Для аппроксимационной задачи использовался решатель LP. Во всех 20 примерах оптимальное решение исходной и аппроксимационной задач совпали. Построить пример, на котором решения этих задач отличались бы друг от друга, пока не удалось.

Заключение

Предложенный подход расширяет возможности анализа инвестиционных проектов и позволяет эффективнее использовать современные методы решения задач при поиске оптимальных расписаний выполнения работ. Кроме проведения экспериментальных расчетов с эволюционными и другими алгоритмами построения приближённого решения, планируется адаптировать точный алгоритм, основанный на схеме динамического программирования. Открытым остаётся вопрос о полиномиальной разрешимости задачи построения оптимального расписания при достаточном собственном финансировании проекта.

ЛИТЕРАТУРА

1. *Fazar W.* The origin of PERT // The Controller. 1962. No. 30. P. 598–621.
2. *Kelley J. E.* Critical-path planning and scheduling: Mathematical basis // Oper. Res. 1961. V. 9. No. 3. P. 296–320.
3. *Alan A., Pritsker B., Watters L. J., and Wolfe P. M.* Multiproject scheduling with limited resources: A zero-one programming approach // Management Science. 1969. V. 16. No. 1. P. 93–108.
4. *Blazewicz J., Lenstra J. K., and Rinnooy Kan A. H. G.* Scheduling subject to resource constraints: classification and complexity // Discr. Appl. Math. 1983. V. 5. No. 1. P. 11–24.
5. *Russell A. H.* Cash flows in networks // Management Science. 1970. V. 16. No. 5. P. 357–373.
6. *Servakh V. V.* A dynamic algorithm for some project management problems // Proc. Int. Workshop Discrete Optimization Methods in Scheduling and Computer-Aided Design (Minsk, September 5–6, 2000). Minsk: Inst. Eng. Cybern. NAS Belarus, 2000. P. 90–92.
7. *Сервах В. В., Щербинина Т. А.* О сложности одной задачи календарного планирования со складируемыми ресурсами // Вестн. НГУ. Сер. матем., мех., информ. 2008. Т. 8. № 3. С. 105–112.
8. *Мартынова Е. А., Сервах В. В.* О задаче календарного планирования проектов с использованием кредитов // Автоматика и телемеханика. 2012. № 3. С. 107–116.
9. *Казаковцева Е. А., Сервах В. В.* Сложность задачи календарного планирования с кредитами // Дискретн. анализ и исслед. опер. 2015. Т. 22. № 4. С. 35–49.
10. *Гимади Э. Х., Залюбовский В. В., Севастьянов С. В.* Полиномиальная разрешимость задач календарного планирования со складируемыми ресурсами и директивными сроками // Дискретн. анализ и исслед. опер. Сер. 2. 2000. Т. 7. № 1. С. 9–34.
11. *Гимади Э. Х., Гончаров Е. Н., Штепа А. А.* Быстрый алгоритм вычисления нижней оценки для решения задачи ресурсно-календарного планирования с тестированием на примерах библиотеки PSPLIB // Тр. ИММ УрО РАН. 2021. Т. 27. № 1. С. 22–36.
12. *Гончаров Е. Н.* Алгоритм локального поиска для задачи календарного планирования с ограниченными ресурсами // Дискретн. анализ и исслед. опер. 2022. Т. 29. № 4. С. 15–37.

13. *Hazir Ö., Haouari M., and Erel E.* Robust optimization for the discrete time-cost tradeoff problem with cost uncertainty // C. Schwindt and J. Zimmermann (eds.). Handbook on Project Management and Scheduling. V. 2. Cham: Springer, 2015. P. 865–874.
14. *Möhring R. H.* Minimizing costs of resource requirements in project networks subject to a fixed completion time // Oper. Res. 1984. V. 32. No. 1. P. 89–120.
15. *Романова А. А.* Минимизация стоимости расписания проекта с возобновимыми ресурсами переменной стоимости // Материалы XII Междунар. школы-симпозиума АМУР. Симферополь, 2018. С. 392–396.
16. *Rodrigues S. B. and Yamashita D. S.* An exact algorithm for minimizing resource availability costs in project scheduling // Europ. J. Oper. Res. 2010. V. 206. No. 3. P. 562–568.
17. *Fu F.* Integrated scheduling and batch ordering for construction project // Appl. Math. Modelling. 2014. V. 38. No. 2. P. 784–797.
18. *Zoraghi N., Shahsavar A., and Niaki S.* A hybrid project scheduling and material ordering problem: Modeling and solution algorithms // Appl. Soft Computing. 2017. V. 58. P. 700–713.
19. *Kononov A. V. and Lin B. M. T.* On relocation problems with multiple working crews // Discret. Optim. 2006. V. 3. No. 4. P. 366–381.
20. *Sevastyanov S. V., Lin B. M. T., and Huang H.-L.* Tight complexity analysis of the relocation problem with arbitrary release dates // Theor. Comput. Sci. 2011. V. 412. No. 35. P. 4536–4544.
21. *Kaplan E. H.* Relocation models for public housing redevelopment programs // Environment and Planning B: Urban Analytics and City Science. 1986. V. 13. No. 1. P. 5–19.
22. *Kaplan E. H. and Amir A.* A fast feasibility test for relocation problems // Europ. J. Oper. Res. 1988. V. 35. No. 2. P. 201–205.
23. *Hanzálek Z. and Šůcha P.* Time symmetry of resource constrained project scheduling with general temporal constraints and take-give resources // Ann. Oper. Res. 2017. V. 248. P. 209–237.
24. *Okubo H., Miyamoto T., Yoshida S., et al.* Project scheduling under partially renewable resources and resource consumption during setup operations // Computers & Industrial Engineering. 2015. V. 83. P. 91–99.
25. *Watermeyer K. and Zimmermann J.* A branch-and-bound procedure for the resource-constrained project scheduling problem with partially renewable resources and general temporal constraints // OR Spectrum. 2020. V. 42. P. 427–460.
26. *Hartmann S. and Briskorn D.* An updated survey of variants and extensions of the resource-constrained project scheduling problem // Europ. J. Oper. Res. 2022. V. 297. No. 1. P. 1–14.
27. Handbook on Project Management and Scheduling. C. Schwindt and J. Zimmermann (eds). Cham: Springer, 2015. 1401 p.
28. *Gu H., Schutt A., Stuckey P. J., et al.* Exact and heuristic methods for the resource-constrained net present value problem // C. Schwindt and J. Zimmermann (eds). Handbook on Project Management and Scheduling. V. 1. Cham: Springer, 2015. P. 299–318.
29. *Leyman P. and Vanhoucke M.* A new scheduling technique for the resource constrained project scheduling problem with discounted cash flows // Intern. J. Production Res. 2015. V. 53. No. 9. P. 2771–2786.
30. *Thiruvady D., Wallace M., Gu H., and Schutt A.* A lagrangian relaxation and ACO hybrid for resource constrained project scheduling with discounted cash flows // J. Heuristics. 2014. V. 20. P. 643–676.

31. *Vanhoucke M.* A scatter search heuristic for maximising the net present value of a resource-constrained project with fixed activity cash flows // Intern. J. Production Res. 2010. V. 48. No. 7. P. 1983–2001.
32. *Fink A. and Homberger J.* An ant-based coordination mechanism for resource-constrained project scheduling with multiple agents and cash flow objective // Flex. Serv. Manuf. J. 2013. V. 25. P. 94–121.
33. *Leyman P. and Vanhoucke M.* Capital- and resource-constrained project scheduling with net present value optimization // Europ. J. Oper. Res. 2017. V. 256. No. 3. P. 757–776.
34. *Leyman P., Van Driessche N., Vanhoucke M., and De Causmaecker P.* The impact of solution representations on heuristic net present value optimization in discrete time/cost trade-off project scheduling with multiple cash flow and payment models // Computers & Oper. Res. 2019. V. 103. P. 184–197.
35. *Leyman P. and Vanhoucke M.* Payment models and net present value optimization for resource constrained project scheduling // Computers & Industrial Engineering. 2016. V. 91. P. 139–153.
36. *Shahsavar M., Niaki S. T. A., and Naja A. A.* An efficient genetic algorithm to maximize net present value of project payments under inflation and bonus–penalty policy in resource investment problem // Adv. Engineering Software. 2010. V. 41. No. 7. P. 1023–1030.
37. *Waligóra G.* Discrete-continuous project scheduling with discounted cash inflows and various payment models — a review of recent results // Ann. Oper. Res. 2014. V. 213. P. 319–340.
38. *Tirkolaee E. B., Goli A., Hematian M., et al.* Multi-objective multi-mode resource constrained project scheduling problem using pare-to-based algorithms // Computing. 2019. V. 101. No. 11. P. 547–570.
39. *Khoshjahan Y., Najafi A. A., and Afshar-Nadjafi B.* Resource constrained project scheduling problem with discounted earliness-tardiness penalties: Mathematical modeling and solving procedure // Computers & Industrial Engineering. 2013. V. 66. No. 2. P. 293–300.
40. *Malakh S. A. and Servakh V. V.* The problem of planning investment projects with lending // LNCS. 2024. V. 14766. P. 187–198.

REFERENCES

1. *Fazar W.* The origin of PERT. The Controller, 1962, no. 30, pp. 598–621.
2. *Kelley J. E.* Critical-path planning and scheduling: Mathematical basis. Oper. Res., 1961, vol. 9, no. 3, pp. 296–320.
3. *Alan A., Pritsker B., Watters L. J., and Wolfe P. M.* Multiproject scheduling with limited resources: A zero-one programming approach. Management Science, 1969, vol. 16, no. 1, pp. 93–108.
4. *Blazewicz J., Lenstra J. K., and Rinnooy Kan A. H. G.* Scheduling subject to resource constraints: classification and complexity. Discr. Appl. Math., 1983, vol. 5, no. 1, pp. 11–24.
5. *Russell A. H.* Cash flows in networks. Management Science, 1970, vol. 16, no. 5, pp. 357–373.
6. *Servakh V. V.* A dynamic algorithm for some project management problems. Proc. Int. Workshop Discrete Optimization Methods in Scheduling and Computer-Aided Design (Minsk, September 5–6, 2000). Minsk, Inst. Eng. Cybern. NAS Belarus, 2000, pp. 90–92.
7. *Servakh V. V. and Shcherbinina T. A.* O slozhnosti odnoy zadachi kalendarnogo planirovaniya so skladiruemymi resursami [Complexity of some project scheduling problem with non-renewable resources]. Vestnik NSU. Ser. Matematika, Mekhanika, Informatika, 2008, vol. 8, no. 3, pp. 105–112. (in Russian)
8. *Martynova E. A. and Servakh V. V.* On scheduling credited projects. Autom. Remote Control, 2012, vol. 73, no. 3, pp. 508–516.

9. *Kazakovtseva E. A. and Servakh V. V.* Complexity of the project scheduling problem with credits. J. Appl. Industr. Math., 2015, vol. 9, no. 4, pp. 489–496.
10. *Gimadi E. Kh., Zalyubovskiy V. V., and Sevast'yanov S. V.* Polinomial'naya razreshimost' zadach kalendarnogo planirovaniya so skladiruemymi resursami i direktivnymi srokami [Polynomial solvability of scheduling problems with storable resources and directive deadlines]. Diskretn. Analiz i Issled. Oper., 2000, ser. 2, vol. 7, no. 1, pp. 9–34. (in Russian)
11. *Gimadi E. Kh., Goncharov E. N., and Shtepa A. A.* Bystryy algoritm vychisleniya nizhney otsenki dlya resheniya zadachi resursno-kalendarnogo planirovaniya s testirovaniem na primerakh biblioteki PSPLIB [A fast algorithm for finding a lower bound of the solution of the resource-constrained project scheduling problem tested on PSPLIB instances]. Trudy Instituta Matematiki i Mekhaniki UrO RAN, 2021, vol. 27, no. 1, pp. 22–36. (in Russian)
12. *Goncharov E. N.* A local search algorithm for the resource-constrained project scheduling problem. J. Appl. Industr. Math., 2022, vol. 16, no. 4, pp. 672–683.
13. *Hazir Ö., Haouari M., and Erel E.* Robust optimization for the discrete time-cost tradeoff problem with cost uncertainty. C. Schwindt and J. Zimmermann (eds.). Handbook on Project Management and Scheduling, vol. 2, Cham, Springer, 2015, pp. 865–874.
14. *Möhring R. H.* Minimizing costs of resource requirements in project networks subject to a fixed completion time. Oper. Res., 1984, vol. 32, no. 1, pp. 89–120.
15. *Romanova A. A.* Minimizatsiya stoimosti raspisaniya proekta s vozobnovimymi resursami peremennoy stoimosti [Minimizing the schedule cost of a project with renewable resources of variable cost]. Proc. XII Intern. Conf. AMUR, Simferopol, 2018, pp. 392–396. (in Russian)
16. *Rodrigues S. B. and Yamashita D. S.* An exact algorithm for minimizing resource availability costs in project scheduling. Europ. J. Oper. Res., 2010, vol. 206, no. 3, pp. 562–568.
17. *Fu F.* Integrated scheduling and batch ordering for construction project. Appl. Math. Modelling, 2014, vol. 38, no. 2, pp. 784–797.
18. *Zoraghi N., Shahsavar A., and Niaki S.* A hybrid project scheduling and material ordering problem: Modeling and solution algorithms. Appl. Soft Computing, 2017, vol. 58, pp. 700–713.
19. *Kononov A. V. and Lin B. M. T.* On relocation problems with multiple working crews. Discret. Optim., 2006, vol. 3, no. 4, pp. 366–381.
20. *Sevastyanov S. V., Lin B. M. T., and Huang H.-L.* Tight complexity analysis of the relocation problem with arbitrary release dates. Theor. Comput. Sci., 2011, vol. 412, no. 35, pp. 4536–4544.
21. *Kaplan E. H.* Relocation models for public housing redevelopment programs. Environment and Planning B: Urban Analytics and City Science, 1986, vol. 13, no. 1, pp. 5–19.
22. *Kaplan E. H. and Amir A.* A fast feasibility test for relocation problems. Europ. J. Oper. Res., 1988, vol. 35, no. 2, pp. 201–205.
23. *Hanzálek Z. and Šůcha P.* Time symmetry of resource constrained project scheduling with general temporal constraints and take-give resources. Ann. Oper. Res., 2017, vol. 248, pp. 209–237.
24. *Okubo H., Miyamoto T., Yoshida S., et al.* Project scheduling under partially renewable resources and resource consumption during setup operations. Computers & Industrial Engineering, 2015, vol. 83, pp. 91–99.
25. *Watermeyer K. and Zimmermann J.* A branch-and-bound procedure for the resource-constrained project scheduling problem with partially renewable resources and general temporal constraints. OR Spectrum, 2020, vol. 42, pp. 427–460.
26. *Hartmann S. and Briskorn D.* An updated survey of variants and extensions of the resource-constrained project scheduling problem. Europ. J. Oper. Res., 2022, vol. 297, no. 1, pp. 1–14.

27. Handbook on Project Management and Scheduling. C. Schwindt and J. Zimmermann (eds). Cham, Springer, 2015. 1401 p.
28. *Gu H., Schutt A., Stuckey P. J., et al.* Exact and heuristic methods for the resource-constrained net present value problem. C. Schwindt and J. Zimmermann (eds). Handbook on Project Management and Scheduling, vol. 1, Cham, Springer, 2015, pp. 299–318.
29. *Leyman P. and Vanhoucke M.* A new scheduling technique for the resource constrained project scheduling problem with discounted cash flows. Intern. J. Production Res., 2015, vol. 53, no. 9, pp. 2771–2786.
30. *Thiruvady D., Wallace M., Gu H., and Schutt A.* A lagrangian relaxation and ACO hybrid for resource constrained project scheduling with discounted cash flows. J. Heuristics, 2014, vol. 20, pp. 643–676.
31. *Vanhoucke M.* A scatter search heuristic for maximising the net present value of a resource-constrained project with fixed activity cash flows. Intern. J. Production Res., 2010, vol. 48, no. 7, pp. 1983–2001.
32. *Fink A. and Homberger J.* An ant-based coordination mechanism for resource-constrained project scheduling with multiple agents and cash flow objective. Flex. Serv. Manuf. J., 2013, vol. 25, pp. 94–121.
33. *Leyman P. and Vanhoucke M.* Capital- and resource-constrained project scheduling with net present value optimization. Europ. J. Oper. Res., 2017, vol. 256, no. 3, pp. 757–776.
34. *Leyman P., Van Driessche N., Vanhoucke M. and De Causmaecker P.* The impact of solution representations on heuristic net present value optimization in discrete time/cost trade-off project scheduling with multiple cash flow and payment models. Computers & Oper. Res., 2019, vol. 103, pp. 184–197.
35. *Leyman P. and Vanhoucke M.* Payment models and net present value optimization for resource constrained project scheduling. Computers & Industrial Engineering, 2016, vol. 91, pp. 139–153.
36. *Shahsavar M., Niaki S. T. A., and Naja A. A.* An efficient genetic algorithm to maximize net present value of project payments under inflation and bonus–penalty policy in resource investment problem. Adv. Engineering Software, 2010, vol. 41, no. 7, pp. 1023–1030.
37. *Waligóra G.* Discrete-continuous project scheduling with discounted cash inflows and various payment models — a review of recent results. Ann. Oper. Res., 2014, vol. 213, pp. 319–340.
38. *Tirkolaee E. B., Goli A., Hematian M., et al.* Multi-objective multi-mode resource constrained project scheduling problem using pare-to-based algorithms. Computing, 2019, vol. 101, no. 11, pp. 547–570.
39. *Khoshjahan Y., Najafi A. A., and Afshar-Nadjafi B.* Resource constrained project scheduling problem with discounted earliness-tardiness penalties: Mathematical modeling and solving procedure. Computers & Industrial Engineering, 2013, vol. 66, no. 2, pp. 293–300.
40. *Malakh S. A. and Servakh V. V.* The problem of planning investment projects with lending. LNCS, 2024, vol. 14766, pp. 187–198.

СВЕДЕНИЯ ОБ АВТОРАХ

АБРОСИМОВ Михаил Борисович — доктор физико-математических наук, профессор, заведующий кафедрой теоретических основ компьютерной безопасности и криптографии Саратовского национального исследовательского государственного университета имени Н. Г. Чернышевского, г. Саратов. E-mail: **mic@rambler.ru**

МАЛАХ Светлана Александровна — младший научный сотрудник Института математики им. С. Л. Соболева СО РАН, г. Омск. E-mail: **malahsveta@mail.ru**

МОРШИНИН Александр Владимирович — кандидат физико-математических наук, научный сотрудник Института математики им. С. Л. Соболева СО РАН, г. Омск. E-mail: **morshinin.alexander@gmail.com**

ПОДЫМОВ Владислав Васильевич — кандидат физико-математических наук, доцент МГУ имени М. В. Ломоносова, г. Москва. E-mail: **valdus@yandex.ru**

ПОПОВ Владимир Юрьевич — доктор физико-математических наук, доцент, профессор кафедры алгебры и фундаментальной информатики, заведующий отделом интеллектуальных систем и робототехники регионального учебно-научного центра по интеллектуальным системам и информационной безопасности Уральского федерального университета имени первого Президента России Б. Н. Ельцина, г. Екатеринбург. E-mail: **popovvvv@gmail.com**

ПОТТОСИН Юрий Васильевич — кандидат физико-математических наук, доцент, ведущий научный сотрудник Объединенного института проблем информатики Национальной академии наук Беларуси, г. Минск. E-mail: **pott@newman.bas-net.by**

РЫБАЛОВ Александр Николаевич — кандидат физико-математических наук, старший научный сотрудник лаборатории комбинаторных и вычислительных методов алгебры и логики Института математики им. С. Л. Соболева СО РАН, г. Омск. E-mail: **alexander.rybalov@gmail.com**

СЕРВАХ Владимир Вицентьевич — доктор физико-математических наук, старший научный сотрудник Института математики им. С. Л. Соболева СО РАН, г. Омск. E-mail: **svv_usa@rambler.ru**

ТЕРЕБИН Богдан Андреевич — аспирант Саратовского национального исследовательского государственного университета имени Н. Г. Чернышевского, г. Саратов. E-mail: **bogdan.terebin@yandex.ru**