S©MR

СИБИРСКИЕ ЭЛЕКТРОННЫЕ
МАТЕМАТИЧЕСКИЕ ИЗВЕСТИЯ
Siberian Electronic Mathematical Reports
http://semr.math.nsc.ru
ISSN 1813-3304

Том 20, № 2, стр. 144–145 (2023) УДК 510.52
https://doi.org/10.33048/semi.2023.20.??? MSC 68Q17

О СКРЫТЫХ СТРУКТУРАХ БУЛЕВЫХ ФУНКЦИЙ

В.Ю. Попов

Abstract: It is well known that Boolean functions have a number of different hidden structures. Some structures are of significant theoretical interest. In particular, the structures associated with the phase transition reveal important properties of the solution space of Boolean functions. Many structures are extensively used by various heuristic algorithms to prove satisfiability of Boolean functions. Among others, we can mention such structures as backdoors, backbones, blocked clauses, asymmetric tautologies, hyper binary resolvents, strongly connected components, bounded variables. Some structures are important in the context of quantum computing. In particular, we can mention the Boolean hidden shift. Typically, to effectively use hidden structures, we need to find not only the structures themselves, but also find the proper assignments for variables. In this paper, we consider the computational complexity of finding proper assignments for variables of hidden structures. In particular, we consider strongly connected components. It is well known that strongly connected components can be easily found in a Boolean function. Nevertheless, we show that the problem of finding proper assignments for the variables of strongly connected components is computationally hard.

Keywords: computational complexity, Boolean function, hidden structures, strongly connected component.

Popov, V.Yu., On the hidden structures of Boolean functions.

Поступила 1 января 2023 г., опубликована 31 декабря 2023 г.

Введение

Теория булевых функций — обширная область, которая давно и интенсивно развивается и имеет широкий спектр разнообразных приложений [1]. Исследования булевых функций позволили выявить большое количество различных скрытых структур [42], идентификация которых существенно облегчает дальнейший анализ булевых функций и установление их свойств.

Некоторые структуры представляют существенный теоретический интерес. В частности, структуры, которые ассоциированы с фазовым переходом вскрывают важные свойства пространства решений булевых функций (см., например, [3, 4, 5]. Фазовый переход характерен для многих физических систем [6, 7, 8]. Недавно доказанная пороговая гипотеза описывает значительное изменение поведения функций в области фазового перехода [9, 10, 11, 12]. В работах было установлено, что для проблемы выполнимости и проблемы коммивояжера имеется тесная связь между фазовым переходом и явлением замораживания переменных [13, 14, 15, 16], которое позднее было формализовано как одна из скрытых структур: хребты [17, 18, 19]. Хребты объясняют принципиально важную аномалию в поведении булевых функций в области фазового перехода: хотя средное вычислительные затраты на поиск решения для проблемы выполнимости ростут линейно с увеличением количества переменных, для небольшой доли булевых функций наблюдается экспоненциальный рост затрат [20], что обусловлено присутствием хребтов в соответствующих булевых функциях [18].

Некоторые структуры важны в контексте квантовых вычислений. В частности, мы можем упомянуть булев скрытый сдвиг [21]. Использование скрытой подгруппы [22] лежит в основе эффективности ряда квантовых алгоритмов [23, 24, 25, 26]. Проблема булева скрытого сдвига является естественным аналогом проблемы скрытой подгруппы для булевых функций [21].

Многие скрытые структуры широко используются различными эвристическими алгоритмами для доказательства выполнимости булевых функций. Среди прочих мы можем упомянуть такие структуры, как бэкдоры [27, 28, 29], хребты [30, 31, 32], блокированные дизъюнкции [33], асимметричные тавтологии [33, 34], гипербинарные резольвенты [33, 35], сильно связанные компоненты [36, 37, 38], ограниченные переменные [33].

Для различных скрытых структур вычислительная сложность выявления их присутствия и нахождения в явном виде существенно отличается. Некоторые скрытые структуры сравнительно легко обнаружить и найти в явном виде. Для некоторых структур вопрос о присутствии этих структур в булевой функции является вычислительно трудным. К числу вычислительно трудных скрытых структур относятся бэкдоры и хребты. Вопрос о существовании бэкдоров исследовался с точки зрения

параметрической сложности [39]. Хотя проблема существования сильных бэкдоров для хорновских бэкдоров и бэкдоров в 2-конъюнктивной нормальной форме разрешима с фиксированным параметром, проблема существования слабых бэкдоров является $W[2]$-трудной для этих бэкдоров [39]. Без фиксации параметра обе проблемы являются **NP**-полными [39]. Для более широкого класса бэкдоров в работе [40] показано, что проблема существования бэкдоров является одновременно **NP**-трудной и **co-NP**-трудной. В предположении $\mathbf{P} \neq \mathbf{NP}$ в работе [41] показано, что не существует аппроксимационной процедуры, которая бы гарантировала нахождение достаточно большого хребта за полиномиальное время. Зафиксируем некоторое множество булевых функций $\mathcal{F}$ и действительное число β, $0 < \beta < 1$. Обозначим через $L(\mathcal{F}, \beta)$ язык, состоящий из всех булевых функций f, $f \in \mathcal{F}$, таких, что существует хребет S, $S \subseteq V(f)$, удовлетворяющий неравенству $|S| \geq \beta|V(f)|$. Для любого действительного числа β, $0 < \beta < 1$, существует генерируемое за полиномиальное время множество булевых функций $\mathcal{F}$ такое, что язык $L(\mathcal{F}, \beta)$ является **NP**-полным [42].

Обычно для эффективного использования скрытых структур мы нуждаемся в нахождении не только самих структур, но и нахождении правильных значений переменных. Более того, во многих случаях мы точно знаем, что булева функция является выполнимой. В частности, это типично для задач, связанных с защитой информации. Например, при атаке на основе проблемы выполнимости на защищенную аппаратную часть мы знаем, что ключ существует, нам необходимо его найти (см., например, [44]). Восстановление исходных данных для хеш-функции естественным образом предполагает их существование (см., например, [45]). Информация об анализируемой булевой функции может включать не только факт ее выполнимости. Например, при решении задач, связанных с защитой от криптоанализа, можно исходить из наличия некоторых вполне определенных скрытых структур (см., например, [46]). В некоторых случаях мы можем быть проинформированы даже о способе сокрытия специфических структур (см., например, [47]). В этих случаях для заведомо выполнимой функции с известными структурами нам необходимо найти конкретные значения переменных этих структур.

Следует отметить, что трудность нахождения значений переменных не всегда напрямую коррелирует с трудностью обнаружения скрытых структур. В частности, в случае хребтов трудной является как задача обнаружения хребтов [41, 42], так и задача нахождения значений для их переменных. С другой стороны, хотя проблема существования сильных бэкдоров для хорновских бэкдоров и бэкдоров в 2-конъюнктивной нормальной форме является **NP**-полной, присутствие сильных бэкдоров обеспечивает проблеме выполнимости для соответствующих функций разрешимость с фиксированным параметром [39].

В этой статье мы рассматриваем вычислительную сложность нахождения правильных значений для переменных скрытых структур. В частности, мы рассматриваем сильно связанные компоненты. Хорошо известно, что сильно связанные компоненты могут быть легко найдены в булевой функции. Тем не менее, мы показываем, что проблема нахождения правильных значений для переменных сильно связанных компонент является вычислительно трудной.

1 Основные определения

Для произвольной булевой функции $f(x_1, x_2, \ldots, x_n)$, зависящей от переменных $x_1, x_2, \ldots, x_n$, будем говорить, что функция f представлена в k-конъюнктивной нормальной форме, если функция f имеет вид

$$\wedge_{i=1}^{m} C_i, \tag{1}$$

где для любого i, $1 \leq i \leq m$, функция C_i является дизъюнкцией не более k литералов, т.е.

$$C_i = \vee_{j=1}^{p_i} u_{i,j}, \tag{2}$$

$$u_{i,j} \in \{x_1, x_2, \ldots, x_n, \neg x_1, \neg x_2, \ldots, \neg x_n\},$$

$p_i \leq k$. Для произвольного литерала u будем предполагать, что u графически равен $\neg\neg u$. Если конкретное значение k не является существенным, то вместо k-конъюнктивной нормальной формы мы будем говорить просто о конъюнктивной нормальной форме.

Графом импликаций [48] булевой функции $f(x_1, x_2, \ldots, x_n)$ в конъюнктивной нормальной форме будем называть ориентированный граф G_f с множеством вершин

$$V_f = \{x_1, x_2, \ldots, x_n, \neg x_1, \neg x_2, \ldots, \neg x_n\}$$

и множеством ребер E_f. Предполагается, что множество E_f содержит ребра $(\neg u_{i,1}, u_{i,2})$ и $(\neg u_{i,2}, u_{i,1})$ тогда и только тогда, когда $C_i = u_{i,1} \vee u_{i,2}$ и литералы $u_{i,1}$ и $u_{i,2}$ соответствуют различным переменным. Кроме пар ребер $(\neg u_{i,1}, u_{i,2})$ и $(\neg u_{i,2}, u_{i,1})$, множество E_f ребер не содержит. В частности, дизъюнкции, содержащие больше двух литералов или только один литерал, в формировании множества ребер не участвуют.

Ориентированный граф называется сильно связным, если для любых двух вершин u и v существуют путь по ребрам из вершины u в вершину v и путь из вершины v в вершину u [49]. Пусть $G = (V, E)$ — ориентированный граф с множеством вершин V и множеством ребер E. Мы можем определить отношение эквивалентности на множестве вершин, полагая, что вершины u и v из множества V эквивалентны, если существуют путь по ребрам из вершины u в вершину v и путь из вершины v в вершину u. Это отношение задает множество

$$\{V_i \mid 1 \leq i \leq n\}$$

попарно различных классов эквивалентности. Рассмотрим множество

$$\{G_i = (V_i, E_i) \mid 1 \leq i \leq n, E_i = \{(u, v) \in E \mid u \in V_i, v \in V_i\}\}$$

подграфов графа G. Каждый граф G_i является сильно связным и не является собственным подграфом никакого сильно связного подграфа графа G. Следуя [49], графы G_i будем называть сильно связными компонентами графа G.

2 Вычислительная сложность нахождения значений переменных

Нахождение сильно связных компонент представляется весьма полезным при анализе булевых функций. По построению сильно связные компоненты определяют множества литералов, принимающих одно и то же значение. Поэтому, заменяя множество переменных на одну переменную, можно существенно снизить размерность пространства перебора. Кроме того, нахождение сильно связных компонент существенно повышает эффективность ряда других методов анализа булевых функций. В частности, это относится к методу зондирования при помощи неудачного литерала (см. [33, 50, 51]). Для нахождения сильно связных компонент произвольного ориентированного графа G хорошо известен ряд эффективных алгоритмов [49, 52, 53]. Учитывая легкость обнаружения, сильно связные компоненты широко используются при анализе булевых функций (см., например, [54, 55, 56]). Однако следует отметить, что для различных скрытых структур их обнаружение имеет существенно различные последствия для дальнейшего анализа булевых функций. Например, обнаружение некоторых бэкдоров дает сравнительно эффективный алгоритм для полного решения проблемы выполнимости для соответствующих функций [39]. Преобразование булевых функций к конъюнктивной нормальной форме, содержащей дизъюнкции из одного литерала, автоматически устанавливает значения соответствующих литералов. С этой точки зрения сильно связные компоненты не столь эффективны: их обнаружение не ведет к непосредственному нахождению значений каких-либо переменных.

В общем случае поиск значений переменных сильно связных компонент может развиваться в двух направлениях: доказательство того, что переменные могут принимать единственное значение; доказательство того, что для переменных сильно связной компоненты возможно свободное назначение, т.е. выполнимость булевой функции не зависит от выбора общего значения эквивалентных переменных сильно связной компоненты. В первом случае переменные сильно связной компоненты являются по определению частью хребта. Соответственно, задача нахождения их значений является вычислительно трудной [43]. Решение вопроса о свободном назначении тоже является **NP**-трудной задачей.

Предложение 1. *Задача выяснения возможности свободного назначения для эквивалентных переменных сильно связной компоненты является* **NP***-трудной.*

Доказательство. Для доказательства предложения достаточно рассмотреть булеву функцию

$$h(x) \wedge (x \vee f), \tag{3}$$

где $h(x)$ — сильно связная компонента, содержащая переменную x, а f — произвольная функция, не имеющая общих переменных с $h(x)$. Если функция f выполнимой не является, то выполнимость функции (3) требует, чтобы выполнялось равенство $x = 1$. Соответственно, в этом случае свободное назначение для эквивалентных переменных сильно связной компоненты $h(x)$ невозможно. Если функция f является выполнимой, то мы можем считать выбор значения переменной x свободным: это ограничивает выбор значений для переменных функции f, но не выводит нас за рамки решения проблемы выполнимости для функции (3). Таким образом, возможность свободного назначения для переменной x равносильна выполнимости булевой функции f. □

Заметим, что утверждение предложения 1, как и результат [43], не являются существенным препятствием для поиска значений переменных сильно связных компонент, поскольку эти результаты не выводят нас за рамки трудности исходной задачи: мы просто одну трудную задачу сводим к другой, имеющей аналогичную трудность. Однако в контексте использования конкретного решателя решение вопроса о свободном назначении может быть значительно труднее, чем вопроса выполнимости исходной функции. Например, хорошо известно, что близкие по виду функции $(\neg x \vee y) \wedge (x \vee \neg y)$ и $(x \vee y) \wedge (x \vee \neg y)$ имеют существенно различные свойства (см., например, [57]).

Предполагая свободу выбора некоторых переменных, мы, вообще говоря, ограничиваем свободу выбора других переменных, что может иметь существенно различные последствия. Например, если в функции $x \vee f$, где x — переменная, а f — нетривиальная функция, для переменной x допустить свободный выбор, то возможность свободного выбора переменных функции f будет ограничена. Однако это ограничение — лишь сведение исходной задачи к подзадаче. Если же допустить свободный выбор для переменной x в функции $f \to x$, где x — переменная, а f — нетривиальная функция, то нам потребуется рассматривать выполнимость функции $\neg f$. Структура функции $\neg f$ может существенно отличаться от структуры функции f, что для конкретного решателя может сделать допущение свободного выбора для переменной x неприемлемым. Поэтому в некоторых случаях элиминация импликации — допущение свободного выбора для переменной, стоящей в заключении импликации — считается недопустимой.

При запрете на элиминацию импликации задача выяснения возможности свободного назначения для эквивалентных переменных сильно связной компоненты становится, вообще говоря, существенно труднее проблемы выполнимости булевой функции. В частности, трудность этой проблемы может быть оценена классом **DP**, определяемым как множество языков $L = L_1 \cap L_2$, где $L_1 \in \mathbf{NP}$, $L_2 \in \mathbf{co-NP}$ (см. [58], пункт 17.1).

Предложение 2. *При запрете на элиминацию импликации задача выяснения возможности свободного назначения для эквивалентных переменных сильно связной компоненты является* **DP***-трудной.*

Доказательство. Доказательство предложения будет основано на сведении от следующей хорошо известной **DP**-полной проблемы (см., например, [58], теорема 17.1).

SAT-UNSAT.

Дано: *Булевы функции f и g, каждая из которых представлена в конъюнктивной нормальной форме, причем каждая дизъюнкция содержит по три литерала.*

Вопрос: *Верно ли, что функция f выполнима, а функция g выполнимой не является?*

Рассмотрим произвольные булевых функции f и g такие, что f и g не имеют общих переменных, каждая из функций f и g представлена в конъюнктивной нормальной форме. Для булевых функции f и g рассмотрим булеву функцию

$$s(f, g) = h(x, y) \wedge (x \vee f) \wedge (g \to y) \tag{4}$$

такую, что $h(x, y)$ — сильно связная компонента, содержащая эквивалентные переменные x и y, функции f, g и h попарно не имеют общих переменных. Для выполнимости функции (4) сильно связная компонента $h(x, y)$ требует лишь выбора одинаковых значений для эквивалентных переменных x и y. Если функция f выполнимой не является, то выполнимость функции (4) требует, чтобы выполнялось равенство $x = 1$. Соответственно, в этом случае свободное назначение для эквивалентных переменных сильно связной компоненты $h(x, y)$ невозможно. Если функция f является выполнимой, то мы можем считать выбор значения переменной x свободным: это ограничивает выбор значений для переменных функции f, но не выводит нас за рамки решения проблемы выполнимости для функции (3). Аналогично, если функция g является выполнимой, то выбор значения переменной y ограничен значением g: поиск опровергающего набора значений для функции g может вести к существенному усложнению поиска. Если функция g не является выполнимой, то выбор значения переменной y можно считать свободным. Учитывая то, что в проблеме SAT-UNSAT множества переменных функций f и g можно считать не пересекающимися, при запрете на элиминацию импликации задача выяснения возможности свободного назначения для

эквивалентных переменных сильно связной компоненты требует решения проблемы SAT-UNSAT, т.е. является **DP**-трудной. □

Хотя в общем случае поиск значений для переменных сильно связных компонент является вычислительно трудной задачей, во многих случаях мы можем располагать информацией о значениях некоторых переменных. Например, значения могут быть выявлены в результате анализа другими методами, знание значений может быть предположением при поиске грубой силой. Однако знание значений некоторых переменных не означает, что их подстановка в булеву функцию является оптимальным действием для дальнейшего анализа: в результате подстановки, вообще говоря, может получиться более трудная для анализа функция. При анализе булевых функций некоторые преобразования направлены на формирование скрытых структур заданного типа в явном виде, а некоторые используются для удаления соответствующих структур. Это обусловлено тем, что присутствие многих скрытых структур может быть как полезным, так и вредным. Ярким примером такой скрытой структуры являются блокированные дизъюнкции. В некоторых случаях элиминация блокированных дизъюнкций ведет к существенному повышению производительности решателя [59, 60, 61]. В других случаях добавление блокированных дизъюнкций повышает производительность решателя [62, 63]. Более того, добавление блокированных дизъюнкций может приводить к экспоненциальному уменьшению кратчайшего опровержения формулы [64, 65]. Таким образом, для некоторой функции f наличие блокированных дизъюнкций может быть полезно, а для некоторой функции g присутствие блокированных дизъюнкций может быть нежелательным. Соответственно, преобразование, например, булевой функции $f \wedge g$ потребует подхода, основанного на встречном использовании добавления и элиминации. Заметим, что выработка оптимальной стратегии для блокированных дизъюнкций является **NP**-трудной [66]. Сильно связные компоненты относятся к числу таких структур. С одной стороны, элиминация сильно связных компонент позволяет значительно сократить пространство решений и упрощает перебор. С другой стороны, введение эквивалентных переменных и, соответственно, добавление сильно связных компонент часто используется для упрощения общей структуры булевой функции. В частности, это преобразование используется для получения планарных формул.

В общем случае булева функция f может иметь некоторое подмножество переменных $X = \cup_{i \in I} X_i$, являющееся объединением множеств переменных X_i сильно связных компонент G_i, $i \in I$, графа G_f, для которых возможна элиминация посредством нахождения значений переменных, и некоторое подмножество переменных Y, для которых появление связных компонент не является желательным. Обычно у нас появляется возможность выделять множества X и Y и рассуждать об их свойствах в том случае, когда анализируемая булева функция f является частью более

общей функции g, некоторые свойства которой нам известны. Говоря о том, что для некоторых переменных возможна элиминация посредством нахождения значений переменных, мы подразумеваем, что у нас есть некоторая эффективная процедура, которая за полиномиальное время позволяет находить значения этих переменных в явном виде. Подставляя значение некоторой переменной, мы подставляем значения и эквивалентных переменнных. Поэтому мы полностью элиминируем некоторую сильно связную компоненту, поскольку компоненты состоят только из эквивалентных переменных. Однако при этом преобразуется вся булева функция f. В результате такого преобразования могут появиться новые сильно связные компоненты или новые переменные в уже существующих компонентах. В некоторых случаях это ведет к существенному упрощению или даже полному решению задачи выполнимости, в некоторых к нежелательным последствиям. Исходя из этого, представляет интерес следующая проблема максимальной подстановки значений переменных сильно связных компонент.

MS.

Дано: *Булева функция f с множеством переменных V. Множество $X = \cup_{i\in I} X_i \subseteq V$ переменных сильно связных компонент G_i, $i \in I$, графа G_f, для которых может быть найдено значение. Множество переменных $Y \subseteq V$. Натуральное число n.*

Вопрос: *Существует ли множество $Z = \cup_{j\in J\subseteq I} X_j \subseteq X$ такое, что $|Z| \geq n$, при подстановке значений переменных из множества Z ни одна переменная множества Y не попадает в сильно связные компоненты графа G_f?*

Предложение 3. *Проблема* MS *является* **NP**-*полной.*

Доказательство. Предположим, что у нас есть некоторое множество Z, являющееся потенциальным решением проблемы MS для булевой функции f. Тогда мы можем за полиномиальное время найти значения всех переменных из множества Z, подставить найденные значения в функцию f и осуществить упрощение: если литерал принимает значение 0, то литерал удаляется из дизъюнкции; при единичном значении литерала удаляется вся дизъюнкция. После этого мы можем, например, используя алгоритм [49], вычислить сильно связные компоненты полученной булевой функции и выяснить, принадлежат ли переменные множества Y какой-либо сильно связной компоненте. Таким образом, проблема MS принадлежит классу **NP**.

Для доказательства трудности проблемы MS мы рассмотрим проблему точного покрытия 3-множествами, которую можно сформулировать следующим образом.

X3C.

Дано: *Натуральное число q. Множество U такое, что $|U| = 3q$. Семейство M подмножеств множества U такое, что для любого $S \in M$ выполняется равенство $|S| = 3$.*

Вопрос: *Верно ли, что семейство M содержит точное покрытие множества U, т.е. такое семейство $M' \subseteq M$, что $U = \cup_{S \in M'} S$ и $|M'| = q$?*

Хорошо известно, что проблема X3C является **NP**-полной (см., например, [58], следствие из теоремы 9.9). Для произвольных исходных данных (q, U, M) проблемы X3C определим некоторую булеву функцию $f(q, U, M)$. Пусть для некоторого натурального числа p выполняется равенство

$$M = \{M_1, M_2, \ldots, M_p\}. \tag{5}$$

Без ограничения общности мы можем предполагать, что $p > 1$, $q > 1$. Над множеством переменных $X \cup Y$, где

$$Y = \{y[1], y[2]\} \tag{6}$$

$$X = \{x[M_s, i] \mid 1 \leq s \leq p, 1 \leq i \leq p^3\} \cup \{x[M_s, M_t] \mid 1 \leq s \leq p, 1 \leq t \leq p, s \neq t\}, \tag{7}$$

рассмотрим следующие булевы функции:

$$g_s = (\neg x[M_s, p^3] \vee x[M_s, 1]) \wedge (\wedge_{1 \leq i < p^3}(\neg x[M_s, i] \vee x[M_s, i+1])) \wedge (\wedge_{s \neq t, M_s \cap M_t \neq \emptyset, 1 \leq t \leq p}(\neg x[M_s, 1] \vee x[M_s, M_t]) \wedge (\neg x[M_s, M_t] \vee x[M_s, 1])), \tag{8}$$

$$h_{s,t} = (\neg y[1] \vee y[2] \vee x[M_s, M_t] \vee x[M_t, M_s]) \wedge (\neg y[2] \vee y[1] \vee x[M_s, M_t] \vee x[M_t, M_s]), \tag{9}$$

$$f(q, U, M) = (\wedge_{1 \leq s \leq p} g_s) \wedge (\wedge_{s \neq t, 1 \leq s \leq p, 1 \leq t \leq p} h_{s,t}). \tag{10}$$

Согласно определениям (6) и (7), множества X и Y не пересекаются. Непосредственной проверкой можно убедиться, что множество переменных V булевой функции $f(q, U, M)$ равно $X \cup Y$. Используя равенства (8) – (10), легко понять, что функции g_s задают сильно связные компоненты. Соответственно,

$$X = \cup_{1 \leq s \leq p} X_s, \tag{11}$$

$$X_s = \{x[M_s, i] \mid 1 \leq i \leq p^3\} \cup \{x[M_s, M_t] \mid 1 \leq t \leq p, s \neq t, M_s \cap M_t \neq \emptyset\}. \tag{12}$$

Будем предполагать, что переменные множества X могут принимать только значение 0. Пусть $n = p^3 q$.

Предположим, что существует множество Z такое, что $|Z| \geq n$, при подстановке значений переменных из множества Z ни одна переменная множества Y не попадает в сильно связные компоненты. Из соотношения (11) следует, что

$$Z = X_{j_1} \cup \cdots \cup X_{j_k}, \tag{13}$$

где $k \leq p$. Непосредственно из соотношения (12) следует, что для любого r, $1 \leq r \leq k$ выполняется неравенство $|X_{j_r}| \leq P^3 + p$. В силу соотношения (13) отсюда вытекает неравенство $|Z| \leq k(P^3 + p)$. Поэтому в силу соотношений $n = p^3 q$, $|Z| \geq n$ выполняется неравенство $p^3 q \leq k(P^3 + p)$. Следовательно, $k \geq q$. Так как при подстановке значений переменных из множества Z ни одна переменная множества Y не попадает в сильно связные компоненты, для любых s и t таких, что $s \neq t$, $M_s \cap M_t \neq \emptyset$, $1 \leq s \leq p$, $1 \leq t \leq p$, либо $X_s \not\subseteq Z$, либо $X_t \not\subseteq Z$. Следовательно, семейство M содержит точное покрытие множества U. В частности, мы можем взять в качестве M' множество

$$M_{j_1} \cup \cdots \cup M_{j_k}.$$

Предположим теперь, что семейство M содержит точное покрытие множества U. В силу соотношения (5) мы можем полагать, что

$$M' = M_{j_1} \cup \cdots \cup M_{j_q}.$$

Тогда в качестве Z мы можем взять множество

$$X_{j_1} \cup \cdots \cup X_{j_q}.$$

Таким образом, семейство M содержит точное покрытие множества U тогда и только тогда, когда для функции $f(q, U, M)$ существует ли множество $Z =\subseteq X$ такое, что $|Z| \geq n$ и при подстановке значений переменных из множества Z ни одна переменная множества Y не попадает в сильно связные компоненты. Отсюда в силу **NP**-полноты проблемы X3C вытекает **NP**-трудность проблемы MS. □

3 Машины Тьюринга для класса TFNP

Результаты предложений 1 и 2 показывают, что нахождение значений для переменных сильно связных компонеет может быть не менее трудным, чем решение проблемы выполнимости. Более того, некоторое усложнение условия анализа булевой функции может сделать задачу нахождения значений переменных существенно более трудной, чем прямое решение проблемы выполнимости. Это согласуется с ранее известными результатами, полученными для других скрытых структур [39, 40, 41, 42, 43]. Однако все эти результаты, за исключением [43], относятся к случаю, когда анализируемая функция теоретически может и не выполняться, т.е. не имеют силы для класса **TFNP**, состоящего из проблем, которые всегда имеют решение.

Определение класса **TFNP** опирается на класс **FNP**. Обычно определение класса **FNP** вводится через определение класса **NP** на основе проверочного отношения. Согласно [67], проверочное отношение — это бинарное отношение $R \subseteq \Sigma^* \times \Pi^*$ для некоторых алфавитов Σ и Π. Произвольному проверочному отношению R можно поставить в соответствие язык $L_R = \{x\#y \mid R(x, y)\}$ над алфавитом $\Sigma \cup \Pi \cup \{\#\}$, где $\# \notin \Sigma$. Говорят, что отношение R вычислимо за полиномиальное время,

если $L_R \in \mathbf{P}$ [67]. Язык L над алфавитом Σ принадлежит классу **NP** тогда и только тогда, когда существует натуральное число k и вычислимое за полиномиальное время проверочное отношение R такие, что для всех $x \in \Sigma^*$ слово x принадлежит языку L тогда и только тогда, когда существует y, удовлетворяющее условиям $|y| \leq |x|^k$ и $R(x, y)$ [67]. Для произвольного языка $L \in \mathbf{NP}$ мы можем определить проблему FL, требующую на вход $x \in \Sigma^*$ ответить "Нет", если $x \notin L$, и найти $y \in \Pi^*$, если $x \in L$. Соответственно, класс **FNP** можно определить как множество $\{FL \mid L \in \mathbf{NP}\}$ (см. [58], глава 10). Формально класс **TFNP** определяется как подкласс класса **FNP**, состоящий из всех проблем имеющих решение для любых исходных данных (см. [58], глава 10).

Традиционно задачи криптографии и большинства других приложений, связанных с защитой информации, предполагают, что решение имеется всегда, т.е. относятся к классу **TFNP**. Соответственно, для таких задач на сегодняшний день известен лишь результат [43]. Хотя результат [43] получен только для хребтов, он может быть распространен и на сильно связные компоненты, и на ряд других скрытых структур. Однако результат [43] справедлив лишь в предположении $\mathbf{P} \neq \mathbf{NP} \cap \mathbf{co\text{-}NP}$, т.е. даже доказательство неравенства $\mathbf{P} \neq \mathbf{NP}$ не гарантирует трудности нахождения значений переменных в классе **TFNP**. Отсутствие более сильного обоснования для криптографических задач обусловлено тем, что большинство методов доказательства трудности опирается на наличие полных проблем, а для класса **TFNP** полные проблемы неизвестны. Более того, основной гипотезой на сегодняшний день является предположение о том, что класс **TFNP** полных проблем не имеет [68, 69, 70].

Вопрос о существовании в некотором классе полных проблем тесно связан с существованием для этого класса рекурсивно перечислимого множества машин. Для доказательства полноты некоторого языка L необходимо показать, что любой язык из класса сводится к L. Соответственно, любое конструктивное доказательство полноты должно опираться на некоторый способ перечисления языков класса. Рекурсивные свойства классов вычислительной сложности рассматривались в работах [71, 72, 73, 74]. В частности, в работе [71] (теорема 5) было показано, что для произвольной функции времени T не существует алгоритма, определяющего по произвольной последовательности α, вычислима ли последовательность α за время T. Согласно [73], для произвольной рекурсивной функции T существует мера вычислительной сложности, относительно которой класс вычислительной сложности, задаваемый функцией T, не является рекурсивно перечислимым. В работе [75] было введено понятие рекурсивной представимости класса вычислительной сложности: класс вычислительной сложности R, состоящий их функций, называется рекурсивно представимым, если существует рекурсивное множество индексов I такое, что для любой функции из R множество I содержит по крайней мере один индекс, множество I не содержит индексов функций, которые не принадлежат R. Как показано в работе

[75], не существует алгоритма, определяющего по произвольному классу вычислительной сложности, является ли он рекурсивно представимым. На необходимость эффективного множества машин для исследования структурных вопросов для классов вычислительной сложности **P** и **NP** было обращено внимание в работах [76, 77]: в работе [76] была предложена модель машины с часами, а в работе [77] использовались машины со счетчиком. В работе [78] (см. также [79]) было показано, что если $\mathbf{P} \neq \mathbf{NP}$, то класс $\mathbf{NP}\backslash\mathbf{P}$ не является рекурсивно представимым. Следует отметить, что список классов вычислительной сложности, для которых доказана рекурсивная представимость, сравнительно невелик [80]. Для произвольной машины Тьюринга M обозначим через $L(M)$ язык, распознаваемый машиной M. Как показано в работе [81], в классе $\mathbf{NP} \cap \mathbf{co\text{-}NP}$ существуют полные языки тогда и только тогда, когда существует рекурсивное множество пар недетерминированных полиномиальных машин Тьюринга $\{(M_i, N_i) \mid i \in \mathbb{N}\}$ такое, что $L(M_i) = \overline{L(N_i)}$, $\mathbf{NP} \cap \mathbf{co\text{-}NP} = \{L(M_i) \mid i \in \mathbb{N}\}$. В работе [82] показано, что для класса **UP** существует полный язык тогда и только тогда, когда существует рекурсивно перечислимое множество однозначных недетерминированных машин $\{N_i \mid i \in \mathbb{N}\}$ такое, что $\mathbf{UP} = \{L(N_i) \mid i \in \mathbb{N}\}$. В связи с этими результатами представляет интерес следующее утверждение.

Предложение 4. *Множество всех полиномиальных недетерминированных машин Тьюринга, решающих проблемы из класса* **TFNP**, *не является рекурсивно перечислимым.*

Доказательство. Как обычно, мы полагаем, что произвольная k-ленточная машина Тьюринга $\mathfrak{M}$ для некоторых натуральных чисел n и m задается множеством внутренних состояний считывающего устройства

$$Q = \{q_0, q_1, q_2, \ldots, q_n\}, \tag{14}$$

где q_0 — выделенное начальное состояние, q_1 — допускающее заключительное состояние, q_2 — отвергающее заключительное состояние, алфавитом ленты

$$\Delta = \{a_0, a_1, \ldots, a_m\}, \tag{15}$$

где a_0 — выделенный пустой символ, и отображением перехода

$$\delta : (Q\backslash\{q_1, q_2\}) \times \Delta^k \to Q \times \Delta^k \times \{-1, 0, 1\}^k. \tag{16}$$

Произвольная команда

$$q_{i_1} a_{j_{1,1}} a_{j_{2,1}} \ldots a_{j_{k,1}} \to q_{i_2} a_{j_{1,2}} a_{j_{2,2}} \ldots a_{j_{k,2}} d_1 d_2 \ldots d_k, \tag{17}$$

где $q_{i_1} \in Q\backslash\{q_1, q_2\}$, $q_{i_2} \in Q$, $0 \leq j_{p,s} \leq m$, $d_p \in \{-1, 0, 1\}$, $1 \leq p \leq k$, $1 \leq s \leq 2$, определяется соотношением

$$\delta(q_{i_1}, a_{j_{1,1}}, a_{j_{2,1}}, \ldots, a_{j_{k,1}}) = (q_{i_2}, a_{j_{1,2}}, a_{j_{2,2}}, \ldots, a_{j_{k,2}}, d_1, d_2, \ldots, d_k)$$

по отображению перехода (16). Программа машины $\mathfrak{M}$ состоит из множества всех команд вида (17). Исходя из введенных определений, произвольная машина Тьюринга $\mathfrak{M}$ может быть задана упорядоченной тройкой (Q, Δ, δ), где Q, Δ и δ определяются соотношениями (14) – (16).

Для доказательства предложения достаточно показать, что существует рекурсивное множество полиномиальных недетерминированных машин Тьюринга такое, что его подмножество, состоящее из машинн, распознающих языки из класса **TFNP**, не является рекурсивно перечислимым. Поэтому мы ограничимся рассмотрением недетерминированных машин Тьюринга с полиномиальными часами. Заметим, что в общем случае полиномиальные часы могут быть реализованы несколькими существенно различными способами. В частности, полиномиальные часы могут быть реализованы как подпрограмма программы машины Тьюринга, которая по входу w длины $|w| = n$ на отдельной ленте вычисляет некоторый фиксированный полином $p(n)$, записывая на ленту $p(n)$ единиц. Предполагается, что машина с часами симулирует программу P некоторой машины Тьюринга M: записав на выделенную ленту одну единицу, машина с часами на остальных лентах симулирует выполнение одной команды программы P на входе w. Таким образом, либо выполнение программы P на входе w естественным образом завершается за $p(n)$ шагов, либо машина с часами записывает $p(n)$ единиц и принудительно останавливает выполнение программы P в допускающем состоянии. Возможна реализация полиномиальных часов как внешнего устройства: часы никак не отображаются в программе машины. В данной статье мы будем предполагать, что машина Тьюринга $\mathfrak{C}$ с полиномиальными часами имеет две ленты. Первую ленту машина $\mathfrak{C}$ использует как рабочую. В частности, на первую ленту машина $\mathfrak{C}$ получает вход. Предполагая, что вход является некоторым словом w, длина которого $|w|$ равна n, на вторую ленту перед началом работы размещается $p(n) \geq n$ единиц. Считывающее устройство машины $\mathfrak{C}$ перед началом работы размещается на крайний левый символ входа w на первой ленте и на крайнюю левую единицу на второй ленте. Если считывающее устройство на второй ленте читает единицу, то выполнение любой команды ведет к смещению вправо на второй ленте. Если на второй ленте читается пустой символ, то машина $\mathfrak{C}$ останавливается в допускающем состоянии. Легко понять, что любая машина с полиномиальными часами распознает некоторый язык за полиномиальное время. Кроме того, любой язык, распознаваемый за полиномиальное время, распознается подходящей машиной с полиномиальными часами.

Доказательство предложения будет основано на сведении от проблемы остановки. При доказательстве мы будем придерживаться формулировок книги [83] (см. пункты 8.2, 8.3.1). Обычно проблема остановки формулируется как задача с двумя входами.

Проблема остановки (Halt).

Дано: *Описание $D(\mathfrak{T})$ детерминированной машины Тьюринга $\mathfrak{T}$, вход w.*

Вопрос: *Верно ли, что машина Тьюринга $\mathfrak{T}$ на входе w через конечное число шагов остановится?*

Однако, фиксируя некоторую машину Тьюринга $\mathfrak{T}$, мы можем рассмотреть проблему остановки для отдельной машины $\mathfrak{T}$.

Проблема остановки для машины $\mathfrak{T}$ (Halt($\mathfrak{T}$)).

Дано: *Вход w.*

Вопрос: *Верно ли, что машина Тьюринга $\mathfrak{T}$ на входе w через конечное число шагов остановится?*

Из неразрешимости проблемы Halt [84] и существования универсальной машины Тьюринга [84] следует неразрешимость проблемы Halt($\mathfrak{T}$) для некоторой детерминированной машины $\mathfrak{T}$. Без ограничения общности мы можем предполагать, что машина $\mathfrak{T}$ является одноленточной (см., например, [85], пункты 8.4.2, 8.4.3, 9.2.3). Зафиксируем некоторую одноленточную детерминированную машину Тьюринга $\mathfrak{T}$ с неразрешимой проблемой остановки. Без ограничения общности мы можем полагать, что машина $\mathfrak{T}$ имеет множество состояний считывающего устройства (14) и алфавит ленты (15). Будем полагать, что программа машины $\mathfrak{T}$ задается функцией перехода

$$\tau : (Q\backslash\{q_1, q_2\}) \times \Delta \to Q \times \Delta \times \{-1, 0, 1\} \tag{18}$$

и состоит из команд вида

$$q_{i_1} a_{j_1} \to q_{i_2} a_{j_2} d, \tag{19}$$

где $q_{i_1} \in Q\backslash\{q_1, q_2\}$, $q_{i_2} \in Q$, $0 \leq j_s \leq m$, $d \in \{-1, 0, 1\}$, $1 \leq s \leq 2$, определяемых соотношениями

$$\tau(q_{i_1}, a_{j_1}) = (q_{i_2}, a_{j_2}, d)$$

по функции перехода (18). Можно предполагать, что машина $\mathfrak{T}$, начиная работу в состоянии q_0, либо через конечное число шагов переходит в состояние q_1 и останавливается, либо работает бесконечно. Без ограничения общности можно полагать, что все входы машины $\mathfrak{T}$ линейно упорядочены и образуют последовательность $w_1, w_2, \ldots$, где w_1 — пустое слово. Мы можем полагать, что если $i \geq j$, то $|w_i| \geq |w_j|$. Для произвольного непустого слова $w_t \in \Delta^*$ в алфавите Δ мы будем предполагать, что $w_t = w[t, 1]w[t, 2] \ldots w[t, |w_t|]$, где $w[t, j] \in \Delta$ для всех $1 \leq j \leq |w_t|$.

Для произвольного слова $w_t \in \Delta^*$ определим машину Тьюринга

$$\mathfrak{C}_{w_t} = (Q', \Delta, \delta')$$

с полиномиальными часами, полагая, что

$$Q' = Q \cup \{q'_0, q'_1, q'_2, \ldots, q'_{|w_t|+3}\},$$

где q'_0 — выделенное начальное состояние, q'_{-1} — допускающее заключительное состояние, q_1 — отвергающее заключительное состояние, программа машины $\mathfrak{C}_{w_t}$ состоит из команд

$$q'_{j-1}x1 \to q'_j w[t,j]111, 1 \le j \le |w_t|, t > 1, x \in \Delta, \tag{20}$$
$$q'_{|w_t|}x1 \to q'_{|w_t|+1}x101, t > 1, x \in \Delta, \tag{21}$$
$$q'_0x1 \to q'_1x101, , t = 1, x \in \Delta, \tag{22}$$
$$q'_jxa_0 \to q'_{-1}xa_000, 1 \le j \le |w_t|, t > 1, x \in \Delta, \tag{23}$$
$$q'_0xa_0 \to q'_{-1}xa_000, t = 1, x \in \Delta, \tag{24}$$
$$q'_{|w_t|+1}y1 \to q'_{|w_t|+1}a_0111, y \in \Delta\backslash\{a_0\}, \tag{25}$$
$$q'_{|w_t|+1}a_01 \to q'_{|w_t|+2}a_01 - 11, t > 1, \tag{26}$$
$$q'_{|w_t|+2}a_01 \to q'_{|w_t|+2}a_01 - 11, t > 1, \tag{27}$$
$$q'_{|w_t|+2}y1 \to q'_{|w_t|+3}y1 - 11, t > 1, y \in \Delta\backslash\{a_0\}, \tag{28}$$
$$q'_{|w_t|+3}y1 \to q'_{|w_t|+3}y1 - 11, t > 1, y \in \Delta\backslash\{a_0\}, \tag{29}$$
$$q'_{|w_t|+3}a_01 \to q_0a_0111, t > 1, \tag{30}$$
$$q'_{|w_t|+1}a_01 \to q_0a_0111, t = 1, \tag{31}$$
$$q'_{|w_t|+1}ya_0 \to q'_{-1}a_0a_000, y \in \Delta\backslash\{a_0\}, \tag{32}$$
$$q'_{|w_t|+1}a_0a_0 \to q'_{-1}a_0a_000, t > 1, \tag{33}$$
$$q'_{|w_t|+2}a_0a_0 \to q'_{-1}a_0a_000, t > 1, \tag{34}$$
$$q'_{|w_t|+2}ya_0 \to q'_{-1}ya_000, t > 1, y \in \Delta\backslash\{a_0\}, \tag{35}$$
$$q'_{|w_t|+3}ya_0 \to q'_{-1}ya_000, t > 1, y \in \Delta\backslash\{a_0\}, \tag{36}$$
$$q'_{|w_t|+3}a_0a_0 \to q'_{-1}a_0a_000, t > 1, \tag{37}$$
$$q'_{|w_t|+1}a_0a_0 \to q'_{-1}a_0a_000, t = 1. \tag{38}$$

Кроме того, для произвольной команды вида (20) машины $\mathfrak{T}$ программа машины $\mathfrak{C}_{w_t}$ содержит команды

$$q_{i_1}a_{j_1}1 \to q_{i_2}a_{j_2}1d1, \tag{39}$$
$$q_{i_1}a_{j_1}a_0 \to q'_{-1}a_0a_000. \tag{40}$$

Легко понять, что команды (20) – (22) обеспечивают печать на первой ленте слова w_t и переход в состояние $q'_{|w_t|+1}$. Команды (25) – (31) позволяют стереть первоначальный вход и переводят считывающее устройство в состояние q_0, устанавливая его на крайний левый символ слова w_t. Команды (23), (24), (32) – (38), (40) обеспечивают работу часов. Для каждой комбинации символа состояния q и символа первой ленты a, для которой машина $\mathfrak{C}_{w_t}$ может выполнить какую-либо команду, мы предусматриваем возможность появления символа a_0 на второй ленте, т.е. остановку часов. Команды (39) позволяют на первой ленте симулировать выполнение программы машины $\mathfrak{T}$ на входе w_t в рамках

ограничений часов. Легко понять, что если машина $\mathfrak{T}$ на входе w_t не останавливается, то машина $\mathfrak{C}_{w_t}$ на всех входах через конечное число шагов останавливается в допускающем состоянии q'_{-1} по часам. Если машина $\mathfrak{T}$ на входе w_t останавливается, то, начиная с некоторого достаточно большого входа w, машина $\mathfrak{C}_{w_t}$ на всех входах через конечное число шагов останавливается в отвергающем состоянии q_1. Таким образом, машина $\mathfrak{C}_{w_t}$ относится к классу **TFNP** тогда и только тогда, когда машина $\mathfrak{T}$ на входе w_t не останавливается. Учитывая то, что дополнение языка проблемы остановки не является рекурсивно перечислимым, отсюда вытекает, что множество машин $\mathfrak{C}_{w_t}$ из класса **TFNP** не является рекурсивно перечислимым. □

Заметим, что машины Тьюринга $\mathfrak{C}_{w_t}$, построенные при доказательстве предложения 4, являются детерминированными. Поэтому для класса **TFP** всех всюду определенных проблем, разрешимых за полиномиальное время детерминированными машинами Тьюринга, имеет место следующее утверждение.

Следствие 1. *Для любого полинома $p(n) \geq n$ множество всех решающих проблемы из класса* **TFP** *полиномиальных детерминированных машин Тьюринга с часами, ограниченными полиномом $p(n)$, не является рекурсивно перечислимым.*

Вопрос о том, решает ли машина Тьюринга всюду определенную проблему, весьма близок к равномерной проблеме остановки, которая может быть сформулирована следующим образом.

Равномерная проблема остановки (UHalt).

Дано: *Описание $D(\mathfrak{T})$ детерминированной машины Тьюринга $\mathfrak{T}$.*

Вопрос: *Верно ли, что машина Тьюринга $\mathfrak{T}$ на любом входе через конечное число шагов переходит в заключительную конфигурацию?*

В книге [83] (см. пункт 8.3.4) показано, что проблема UHalt неразрешима для одноленточных машин Тьюринга с одним выделенным заключительным состоянием. Конструкция, рассмотренная при доказательстве предложения 4, позволяет получить аналог этого утверждения для полиномиальных машин Тьюринга.

Следствие 2. *Для любого полинома $p(n) \geq n$ проблема* UHalt *неразрешима на множестве всех полиномиальных детерминированных машин Тьюринга с часами, ограниченными полиномом $p(n)$.*

Доказательство. Если в программу машины $\mathfrak{C}_{w_t}$ добавить команды

$$q_1xy \to q_1xy00, x \in \Delta, y \in \Delta,$$

то машина $\mathfrak{C}_{w_t}$ будет останавливаться на всех входах почасам, если машина $\mathfrak{T}$ с неразрешимой проблемой остановки на входе w_t не останавливается. Если машина $\mathfrak{T}$ на входе w_t не останавливается, то, начиная с некоторого достаточно большого входа w, машина $\mathfrak{C}_{w_t}$ будет работать бесконечно на всех входах. □

Традиционный подход к использованию методов защиты информации предполагает применение некоторых фиксированных алгоритмов, которые известны злоумышленникам. На сегодняшний день в рамках этого подхода известны методы обоснования трудности лишь в предположении $\mathbf{P} \neq \mathbf{NP} \cap \mathbf{co\text{-}NP}$. Утверждение предложения 4 является существенным аргументом в пользу того, что класс **TFNP** не имеет полных проблем, что является принципиальной преградой для более весомого обоснования трудности, чем то, что можно извлечь из предположения $\mathbf{P} \neq \mathbf{NP} \cap \mathbf{co\text{-}NP}$.

Однако утверждение предложения 4 открывает альтернативную возможность обоснования трудности задач, связанных с защитой информации. Если алгоритм защиты не является фиксированным, а выбирается из некоторого множества $M \subseteq \mathbf{TFNP}$, то злоумышленнику, решая задачу взлома защиты, необходимо будет, в частности, решать задачу принадлежности множеству M. С учетом того, что множество **TFNP** не является рекурсивно перечислимым, сложность задачи принадлежности множеству M может быть, вообще говоря, настроена на произвольный уровень вплоть до алгоритмической неразрешимости. При этом высокий уровень надежности системы защиты информации может быть гарантирован даже в том случае, когда индивидуальные задачи из множества M могут быть решены детерминированными полиномиальными алгоритмами, т.е. относятся к классу **TFP**. Соответственно, при таком подходе к построению системы защиты информации трудность взлома сохранится на высоком уровне даже в случае доказательства равенства $\mathbf{P} = \mathbf{NP}$.

Заметим, что повышение сложности множества M в общем случае создает определенные трудности как злоумышленнику, так и защитнику. Однако злоумышленнику необходимо решать задачу принадлежности множеству M, что соответствует проблеме распознавания множества M: в общем случае множество распознаваемых языков равно классу рекурсивных языков **R**. Защитнику необходимо уметь решать лишь задачу генерации множества M, что соответствует проблеме перечисления множества M: в общем случае множество перечислимых языков равно классу рекурсивно перечислимых языков **RE**. Хорошо известно, что $\mathbf{R} \subset \mathbf{RE}$. Поэтому в общем случае защитник имеет существенное преимущество, располагая возможностью выбрать язык из множества $\mathbf{RE} \backslash \mathbf{R}$. Естественно соотношение $\mathbf{R} \subset \mathbf{RE}$ гарантирует защитнику лишь теоретическое преимущество, поскольку с практической точки зрения представляют интерес лишь полиномиальные вычисления. На сегодняшний день известно несколько полиномиальных аналогов класса рекурсивно перечислимых языков (см., например, [72, 86, 87, 88]). В частности, в работе [88] введено понятие полиномиальной перечислимости по итерации. Как показано в работе [88], класс языков, которые полиномиально перечислимы по итерации, является весьма обширным (см. [88], теорема 4.2) и, в частности, включает все **EXPTIME**-полные языки (см. [88],

следствие 4.7), где **EXPTIME** — класс языков, которые распознаваемы за экспоненциальное время. Поэтому защитник за полиномиальное время может генерировать, например, множество M, являющееся некоторой комбинацией **EXPTIME**-полных языков. При этом злоумышленнику придется осуществлять гарантированно экспоненциальное по времени вычисление только для решения задачи принадлежности множеству M. Таким образом, учитывая то, что $\mathbf{P} \neq \mathbf{EXPTIME}$, при использовании такого подхода к шифрованию невозможность взлома за полиномиальное время можно гарантировать даже в случае $\mathbf{P} = \mathbf{NP}$.

Заключение

В данной работе мы установили вычислительную сложность ряда проблем, связанных с нахождением значений переменных сильно связных компонент булевых функций. В частности, мы показали, что задача нахождения значений переменных сильно связных компонент булевых функций не менее трудна, чем решение проблемы выполнимости. Кроме того, мы рассмотрели вопрос вычислительной трудности задач поиска значений переменных в том случае, когда известно, что они существуют. Для этого случая мы показали, что массовая задача нахождения значений переменных для произвольной булевой функции может быть алгоритмически неразрешимой. В частности, установлено, что для любого полинома $p(n) \geq n$ множество всех решающих проблемы из класса **TFP** полиномиальных детерминированных машин Тьюринга с часами, ограниченными полиномом $p(n)$, не является рекурсивно перечислимым.

References

[1] Y. Crama, P. L. Hammer, *Boolean Functions: Theory, Algorithms, and Applications*, Cambridge University Press, Cambridge, 2011.

[2] L. A. Hemaspaandra, D. E. Narváez, *Existence versus exploitation: the opacity of backdoors and backbones*, Progress in Artificial Intelligence, **10** (2021), 297–308.

[3] P. Cheeseman, B. Kanefsky, W. M. Taylor, *Where the really hard problems are*, Proceedings of the 12th international joint conference on Artificial intelligence, **1** (1991), 331–337.

[4] D. Mitchell, B. Selman, H. Levesque, *Hard and easy distributions of SAT problems*, Proceedings of the 10th National Conference on Artificial Intelligence, AAAI, 1992, 459–465.

[5] J. Sleegers, R. Olij, G. van Horn, D. van den Berg, *Where the really hard problems aren't*, Operations Research Perspectives, **7** (2000), 100160.

[6] A. K. Hartmann, M. Weigt, *Phase Transitions in Combinatorial Optimization Problems: Basics, Algorithms and Statistical Mechanics*, WILEY-VCH Verlag GmbH & Co. KGaA, Weinheim, 2005.

[7] M. Mézard, A. Montanari, *Information, Physics, and Computation*, Oxford University Press, New York, 2009.

[8] C. Moore, S. Mertens, *The Nature of Computation*, Oxford University Press, New York, 2011.

[9] J. Ding, A. Sly, N. Sun, *Satisfiability Threshold for Random Regular NAE-SAT*, Communications in Mathematical Physics, **341** (2016), 435–489.
[10] J. Ding, A. Sly, N. Sun, *Proof of the satisfiability conjecture for large k*, Annals of Mathematics, **196** (2022), 1–388.
[11] J. Park, H. T. Pham, *A proof of the Kahn–Kalai conjecture*, Journal of the American Mathematical Society, **37** (2024), 235–243.
[12] B. Park, J. Vondrák, *A Simple Proof of the Nonuniform Kahn – Kalai Conjecture*, SIAM Journal on Discrete Mathematics, **38** (2024).
[13] S. Kirkpatrick, G. Toulouse, *Configuration space analysis of travelling salesman problems*, Journal de Physique, **45** (1985) 1277–1292.
[14] R. Monasson, R. Zecchina, *Entropy of the K-Satisfiability Problem*, Physical review letters, **76** (1996) 3881–3885.
[15] R. Monasson, R. Zecchina, *Statistical mechanics of the random K-satisfiability model*, Physical review E, **56** (1997) 1357–1370.
[16] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, L. Troyansky, *2+p-SAT: Relation of typical-case complexity to the nature of the phase transition*, Random Structures & Algorithms, **15** (1999) 414–435.
[17] J. Schneider, C. Froschhammer, I. Morgenstern, T. Husslein, J. M. Singer, *Searching for backbonesSearching for backbones — an efficient parallel algorithm for the traveling salesman problem*, Computer Physics Communications, **96** (1996) 173–188.
[18] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, L. Troyansky, *Determining computational complexity from characteristic 'phase transitions'*, Nature, **400** (1999) 133–137.
[19] B. Bollobás, C. Borgs, J. T. Chayes, J. H. Kim, D. B. Wilson, *The scaling window of the 2-SAT transition*, Random Structures & Algorithms, **18** (2001) 201–256.
[20] B. Selman, S. Kirkpatrick, *Critical behavior in the computational cost of satisfiability testing*, Artificial Intelligence, **81** (1996) 273–295.
[21] A. M. Childs, R. Kothari, M. Ozols, M. Roetteler, *Easy and Hard Functions for the Boolean Hidden Shift Problem*, 8th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2013), Leibniz International Proceedings in Informatics (LIPIcs), Volume 22, Leibniz-Zentrum für Informatik, Schloss Dagstuhl, 2013, 50–79.
[22] A. M. Childs, W. van Dam, *Quantum algorithms for algebraic problem*, Reviews of Modern Physics, **82** (2010) 1–52.
[23] D. Boneh, R. Lipton, *Quantum cryptanalysis of hidden linear functions*, Lecture Notes in Computer Science, **963** (1995) 424–437.
[24] O. Regev, *Quantum computation and lattice problems*, SIAM Journal on Computing, **33** (2004) 738–760.
[25] G. Kuperberg, *A subexponential-time quantum algorithm for the dihedral hidden subgroup problem*, SIAM Journal on Computing, **35** (2005) 170–188.
[26] S. Hallgren, *Polynomial-time quantum algorithms for Pell's equation and the principal ideal problem*, Journal of the ACM, **54** (2007) 4:1–4:19.
[27] S. Gaspers, N. Misra, S. Ordyniak, S. Szeider, S. Zivný, *Backdoors into heterogeneous classes of SAT and CSP*, Journal of Computer and System Sciences, **85** (2017) 38–56.
[28] M. Samer, S. Szeider, *Backdoor Sets of Quantified Boolean Formulas*, Journal of Automated Reasoning, **42** (2008) 77–97.
[29] S. Szeider, *Matched Formulas and Backdoor Sets*, Journal on Satisfiability, Boolean Modelling and Computation, **6** (2010) 1–12.
[30] T. Al-Yahya, M. E. B. A. Menai, H. Mathkour, *Boosting the Performance of CDCL-Based SAT Solvers by Exploiting Backbones and Backdoors*, Algorithms, **15** (2022) 302.
[31] M. E. B. Menaï, M. Batouche, *A Backbone-Based Co-evolutionary Heuristic for Partial MAX-SAT*, Lecture Notes in Computer Science, **3871** (2006) 155–166.

[32] A. Biere, T. Faller, K. Fazekas, M. Fleury, N. Froleyks, F. Pollitt, *CaDiCaL 2.0*, Lecture Notes in Computer Science, **14681** (2024) 133–152.
[33] A. Biere, M. Järvisalo, B. Kiesl, *Preprocessing in SAT Solving*, Frontiers in Artificial Intelligence and Applications, **336** (2021) 391–435.
[34] M. Kirchweger, S. Szeider, *SAT Modulo Symmetries for Graph Generation and Enumeration*, ACM Transactions on Computational Logic, **25** (2024) 1–30.
[35] M. J. H. Heule, M. Järvisalo, A. Biere, *Revisiting Hyper Binary Resolution*, Lecture Notes in Computer Science, **7874** (2013) 77–93.
[36] M. Heule, M. Järvisalo, A. Biere, *Clause Elimination Procedures for CNF Formulas*, Lecture Notes in Computer Science, **6397** (2010) 357–371.
[37] C. M. Li, *Equivalency reasoning to solve a class of hard SAT problems*, Information Processing Letters, **76** (2000) 75–81.
[38] P. Vukmirović, J. Blanchette, M. J. H. Heule, *SAT-Inspired Eliminations for Superposition*, ACM Transactions on Computational Logic, **24** (2023) 1–25.
[39] N. Nishimura, P. Ragde, S. Szeider, *Detecting backdoor sets with respect to Horn and binary clauses*, Lecture Notes in Computer Science, **3542** (2005) 96–103.
[40] B. Dilkina, C. P. Gomes, A. Sabharwal, *Tradeoffs in the complexity of backdoors to satisfiability: dynamic sub-solvers and learning during search*, Annals of Mathematics and Artificial Intelligence, **70** (2014) 399–431.
[41] P. Kilby, J. Slaney, S. Thiebaux, T. Walsh, *Backbones and backdoors in satisfiability*, Proceedings of the AAAI Conference on Artificial Intelligence, **20** (2005) 1368–1373.
[42] L. A. Hemaspaandra, D. E. Narváez, *Existence versus exploitation: the opacity of backdoors and backbones*, Progress in Artificial Intelligence, **10** (2021) 297–308.
[43] L. A. Hemaspaandra, D. E. Narváez, *The opacity of backbones*, Information and Computation, **281** (2021) 104772.
[44] A. Fowler, S. Mohammed, M. Shihab, T. Broadfoot, P. Beerel, C. Sechen, Y. Makris, *A TRAP for SAT: On the Imperviousness of a Transistor-Level Programmable Fabric to Satisfiability-Based Attacks*, IACR Transactions on Cryptographic Hardware and Embedded Systems, **2025** (2025), 579–603.
[45] F. Lafitte, J. Nakahara Jr., D. Van Heule, *Applications of SAT Solvers in Cryptanalysis: Finding Weak Keys and Preimages*, Journal on Satisfiability, Boolean Modelling and Computation, **9** (2014), 1–25.
[46] J. Shi, G. Liu, C. Li, *SAT-Based Security Evaluation for WARP against Linear Cryptanalysis*, IET Information Security, **2023** (2023), 5323380.
[47] S. Matsui, K. Cai, *Usability aware secret protection with minimum cost*, Nonlinear Analysis: Hybrid Systems, **43** (2021), 101111.
[48] B. Aspvall, M. F. Plass, R. E. Tarjan, *A linear-time algorithm for testing the truth of certain quantified boolean formulas*, Information Processing Letters, **8** (1979), 121–123.
[49] R. Tarjan, *Depth-First Search and Linear Graph Algorithms*, SIAM Journal on Computing, **1** (1972), 146–160.
[50] D. Le Berre, *Exploiting the real power of unit propagation lookahead*, Electronic Notes in Discrete Mathematics, **9** (2001), 59–80.
[51] I. Lynce, J. Marques-Silva, *Probing-based preprocessing techniques for propositional satisfiability*, Proceedings. 15th IEEE International Conference on Tools with Artificial Intelligence, IEEE, Piscataway, 2003.
[52] M. Sharir, *A strong-connectivity algorithm and its applications in data flow analysis*, Computers & Mathematics with Applications, **7** (1981), 67–72.
[53] L. K. Fleischer, B. Hendrickson, A. Pinar, *On Identifying Strongly Connected Components in Parallel*, Lecture Notes in Computer Science, **1800** (2000), 505–511.
[54] R.I. Brafman, *A simplifier for propositional formulas with many binary clauses*, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), **34** (2004), 52–59.

[55] J. E. Reeves, M. J. H. Heule, R. E. Bryant, *From Clauses to Klauses*, Lecture Notes in Computer Science, **14681** (2024), 110–132.
[56] A. Van Gelder, *Toward leaner binary-clause reasoning in a satisfiability solver*, Annals of Mathematics and Artificial Intelligence, **43** (2005), 239–253.
[57] C.-M. Li, *Equivalent literal propagation in the DLL procedure*, Discrete Applied Mathematics, **130** (2003), 251–276.
[58] C. H. Papadimitriou, *Computational Complexity*, Addison-Wesley Publishing Co. Inc., Boston, 1994.
[59] A. Biere, F. Lonsing, M. Seidl, *Blocked Clause Elimination for QBF*, Lecture Notes in Computer Science, **6803** (2011), 101–115.
[60] M. J. H. Heule, O. Kullmann, V. W. Marek, *Solving and Verifying the Boolean Pythagorean Triples Problem via Cube-and-Conquer*, Lecture Notes in Computer Science, **9710** (2016), 228–245.
[61] M. Järvisalo, A. Biere, M. Heule, *Blocked Clause Elimination*, Lecture Notes in Computer Science, **6015** (2010), 129–144.
[62] M. Fleury, D. Kaufmann, *Life span of SAT techniques*, arXiv:2402.01202v, 2024.
[63] B. Subercaseaux, M. J. H. Heule, *The Packing Chromatic Number of the Infinite Square Grid is 15*, Lecture Notes in Computer Science, **13993** (2023), 389–406.
[64] B. Kiesl, A. Rebola-Pardo, M. J. H. Heule, *Extended Resolution Simulates DRAT*, Lecture Notes in Computer Science, **10900** (2018), 516–531.
[65] O. Kullmann, *On a generalization of extended resolution*, Discrete Applied Mathematics, **96-97** (1999), 149–176.
[66] B. Subercaseaux, *Sometimes Hoarding is Harder than Cleaning: NP-hardness of Maximum Blocked-Clause Addition*, EPiC Series in Computing, **100** (2024), 408–425.
[67] S. Cook, *The P versus NP Problem*, J. Carlson, A. Jaffe, and A. Wiles (Eds.), The Millennium Prize Problems. Providence: American Mathematical Society, 2006. P. 87–104.
[68] N. Megiddo, C. H. Papadimitriou, *On total functions, existence theorems and computational complexity*, Theoretical Computer Science, **81** (1991), 317–324.
[69] P. W. Goldberg, C. H. Papadimitriou, *Towards a unified complexity theory of total functions*, Journal of Computer and System Sciences, **94** (2018), 167–192.
[70] M. Göös, A. Hollender, S. Jain, G. Maystre, W. Pires, R. Robere, R. Tao, *Separations in Proof Complexity and TFNP*, Journal of the ACM, **71** (2024), 1–45.
[71] J. Hartmanis, R. E. Stearns, *On the Computational Complexity of Algorithms*, Transactions of the American Mathematical Society, **117** (1965), 285–306.
[72] P. R. Young, *Toward a Theory of Enumerations*, Journal of the ACM, **16** (1969), 328–348.
[73] F. D. Lewis, *Unsolvability considerations in computational complexity*, STOC '70: Proceedings of the second annual ACM symposium on Theory of computing, Association for Computing Machinery, New York, 1970, 22–30.
[74] A. Borodin, *Computational Complexity and the Existence of Complexity Gaps*, Journal of the ACM, **19** (1972), 158–174.
[75] L. H. Landweber, E. L. Robertson, *Recursive Properties of Abstract Complexity Classes*, Journal of the ACM, **19** (1972), 296–308.
[76] T. Baker, J. Gill, R. Solovay, *Relativizations of the $\mathcal{P} =? \mathcal{NP}$ Question*, SIAM Journal on Computing, **4** (1975), 431–442.
[77] R. E. Ladner, *On the Structure of Polynomial Time Reducibility*, Journal of the ACM, **22** (1975), 155–171.
[78] L. H. Landweber, R. J. Lipton, E. L. Robertson, *On the structure of sets in NP and other complexity classes*, Theoretical Computer Science, **15** (1981), 181–200.
[79] P. Chew, M. Machtey, *A note on structure and looking back applied to the relative complexity of computable functions*, Journal of Computer and System Sciences, **22** (1981), 53–59.

[80] U. Schöning, *A uniform approach to obtain diagonal sets in complexity classes*, Theoretical Computer Science, **18** (1982), 95–103.
[81] W. Kowalczyk, *Some Connections Between Representability of Complexity Classes and the Power of Formal Systems of Reasonin*, Lecture Notes in Computer Science, **176** (1984), 364–369.
[82] J. Hartmanis, L. A. Hemachandra, *Complexity classes without machines: On complete languages for UP*, Theoretical Computer Science, **58** (1988), 129–142.
[83] M. L. Minsky, *Computation: finite and infinite machines*, Prentice-Hall, Englewood Cliffs, 1967.
[84] A. M. Turing, *On Computable Numbers, with an Application to the Entscheidungsproblem*, Proceedings of the London Mathematical Society, series 2, **42** (1936-7), 230–265.
[85] J. E. Hopcroft, R. Motwani, J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Boston, 2000.
[86] A. Selman, *Polynomial time enumeration reducibility*, SIAM Journal on Computing, **7** (1978), 440–457.
[87] J. Hartmanis, Y. Yesha, *Computation times of NP sets of different densities*, Theoretical Computer Science, **34** (1984), 17–32.
[88] L. A. Hemachandra, A. Hoene, D. Siefkes, P. Young, *On sets polynomially enumerable by iteration*, Theoretical Computer Science, **80** (1991), 203–225.

Vladimir Yurievich Popov
Ural Federal University,
Lenin st., 51,
620083, Ekaterinburg, Russia
Email address: popovvvv@gmail.com