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

ДИЗЪЮНКТИВНО-СВЯЗНЫЕ ВАРИАНТЫ ПРОБЛЕМЫ ВЫПОЛНИМОСТИ

В.Ю. Попов

Abstract: The satisfiability problem is one of the most famous computationally hard algorithmic problems. It is well known that the satisfiability problem remains hard even in the restricted version in which Boolean formulas in conjunctive normal form with exactly three distinct literals per clause. However, the problem can be solved in polynomial time for Boolean formulas with exactly two distinct literals per clause. Narrowing the gap between the problems is of fundamental interest. Therefore, it is natural to analyze the complexity of some restricted versions of the satisfiability problem. In this paper, we prove hardness of some clause-connected versions of the satisfiability problem.

Keywords: satisfiability problem, computational complexity, **NP**-complete.

Введение

Проблема выполнимости булевой функции (SAT) является одной из наиболее известных вычислительно трудных алгоритмических проблем. Обычно проблема SAT рассматривается для булевых функций в конъюнктивной нормальной форме. При этом хорошо известно, что проблема 3SAT, в которой допускается в дизъюнкциях ровно по три литерала,

Popov, V.Yu., Clause-connected versions of the satisfiability problem.

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

остается **NP**-полной, а проблема 2SAT, в которой допускается в дизъюнкциях ровно по два литерала, разрешима за полиномиальное время. Сужение зазора между 3SAT и 2SAT представляет фундаментальный интерес и вызывает большое внимание у исследователей (см., например, [1, 2, 3]). Для рассмотрения ограниченных версий проблемы 3SAT имеется большое количество дополнительных важных причин.

Проблема 3SAT использовалась для получения многочисленных результатов по вычислительной трудности. Поэтому доказательство трудности для ограниченного варианта 3SAT во многих случаях позволяет получить важные следствия для других проблем.

Рассмотрим произвольную булеву функцию $f(x_1, x_2, \ldots, x_n)$ в конъюнктивной нормальной форме. Без ограничения общности мы можем полагать, что функция $f(x_1, x_2, \ldots, x_n)$ имеет вид

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

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

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

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

Сведение различных вычислительно трудных проблем к проблема выполнимости и решение их при помощи специально разработанных для SAT программ уже давно рассматривается как эффективный метод решения задач. Однако стандартная процедура приведения произвольной булевой функции к виду (1) является, вообще говоря, экспоненциальной. Даже использование эффективных алгоритмов (см., например, [4]) для приведения к виду (1) накладывает значительную дополнительную вычислительную нагрузку на программу-решатель. Конкретный вид выражений (2) тоже имеет важное значение, поскольку упрощение выражения (2) может оказать существенное влияние на количество переменных, количество и структуру решений. Поэтому при построении сведений прикладных задач к проблеме выполнимости важно заботиться о конкретном виде получаемых булевых функций. При этом, накладывая ограничения на вид булевых функций, необходимо знать, сохраняет ли ограниченный вариант проблемы SAT выразительные способности SAT. С другой стороны при построении сведений задач робототехники (см., например, [5, 6, 7]), планирования для автономных транспортных средств (см., например, [8, 9]), майнинга криптовалюты [10], криптоанализа (см., например, [11, 12]), моделирования квантовых вычислительных процессов (см., например, [13, 14]) и многих других часто булевы функции специального вида возникают естественным образом. Даже если выявление дополнительной структуры не ведет к нахождению полиномиального алгоритма, во многих случаях появляется возможность для создания более эффективной программы-решателя. В последние годы при разработке программ-решателей все большее внимание уделяется

не стандартным формулировкам SAT и 3SAT, а упрощенным вариантам 3SAT (см., например, [15]). Во многих случаях известные специализированные варианты проблемы 3SAT недостаточно хорошо отражают специфику прикладной задачи, что требует введения сложных дополнительных структур и приводит к совместному решению нескольких задач (см., например, [16]). Исследование различных прикладных задач приводит к рассмотрению новых вариантов проблемы 3SAT. При этом существенное внимание уделяется исследованию дополнительных ограничений на проблему 3SAT, обеспечивающих некоторую связность между переменными в различных дизъюнкциях (см., например, [17, 18]). Характерным примером является исследование, представленное в работе [19], где вариант проблемы 3SAT с дополнительным условием на связность переменных решается совместно с другой оптимизационной задачей.

Программы-решатели для вариантов проблемы выполнимости демонстрируют высокую эффективность при решении многих прикладных задач. Однако часто их использование сопряжено с трудностями, обусловленными недостаточным количеством тестовых данных [20], что вызывает потребность генерации булевых функций специального вида (см., например, [21, 22, 23]). Активное использование методов машинного обучения для программ-решателей (см., например, [24, 25, 26]) требует учета специфики не только прикладной области, но и самих методов машинного обучения. Хорошо известно, что интеллектуальные системы обычно медленно обучаются на разреженных данных. С другой стороны данные высокой плотности часто вызывают быстрое переобучение. Кроме того, специфика исходных данных вызывает определенные трудности в применении программ-решателей для криптоанализа (см., например, [27, 28]).

Многие современные криптографические алгоритмы работают с разреженными или плотными данными (см., например, [29, 30, 31]). Соответственно разрабатываются методы криптоанализа, которые ориентируются на поиск специфических закономерностей, связанных с разреженными и плотными данными [32, 33]. Принципиальная возможность нахождения подобных закономерностей основывается на наличии фазового перехода, который характерен не только для алгоритмических проблем, но и для физических систем в целом [34, 35, 36]. Суть фазового перехода в том, что для исходных данных одного и того же размера структура решений может значительно отличаться. Соответственно, трудность проблемы зависит не столько от размера исходных данных, сколько от некоторых их структурных особенностей. Исследования этого феномена тесно связаны с пороговой гипотезой, согласно которой трудность исходных данных зависит от отношения размера исходных данных к количеству переменных, т.е. от степени их плотности. Для проблемы 3SAT это отношение можно рассматривать как отношение $\frac{m}{n}$ количества дизъюнкций m к количеству переменных n [37]. Многие вычислительно трудные проблемы демонстрируют предсказуемое поведение в

зависимости от плотности исходных данных, что существенно облегчает их решение (см., например, [38, 39]). Весьма значительное внимание было уделено изучению пороговой гипотезы для проблемы SAT (см., например, [40, 41]. Обзор современного состояния исследований по пороговой гипотезе дан в работе [42]. В частности, следует отметить, что недавно пороговая гипотеза получила подтверждение как для проблемы выполнимости [43, 44], так и в общем случае [45]. Однако исследования, представленные в работе [46], показывают наличие нетривиальных зависимостей у проблемы выполнимости, которые не могут быть описаны одним лишь подтверждением пороговой гипотезы. В частности, у проблемы 3SAT наблюдается несколько фазовых переходов. Тем не менее, нахождение эффективных методов, позволяющих гибко регулировать связность булевых функций, представляет значительный интерес для разработки методов генерации булевых функций и маскировки для криптографических алгоритмов. В частности, их можно было бы применять для тех значений плотности, для которых общее поведение решений проблемы выполнимости сравнительно предсказуемо. При этом при помощи полиномиальных сведений маскировку на основе булевых функций можно применять не только для сравнительно непопулярных алгоритмов, основанных непосредственно на SAT [47, 48, 49], но и для алгоритмов на целочисленных решетках или схем на основе Classic McEliece.

Пожалуй, наиболее основательно вопрос о требованиях к булевым функциям проработан в области генерации трудных входов для программ-решателей (см., например, [50, 51, 52]). В частности, в работе [50] указывается, что в отличие от случайных булевых функций индустриальные должны демонстрировать наличие определенной структуры связей. При этом одной из ключевых проблем генерации является деградация сложности из-за ограниченности инструментов модификации [50]. В свою очередь, ограниченные возможности модификации обусловлены стремлением сохранить семантику булевых функций, получаемых из экспериментальных данных. Таким образом, необходимы эффективные инструменты, позволяющие осуществлять гибкую модификацию структуры связей в булевой функции, наследуя трудность исходной функции. С другой стороны, для разработки самих решателей необходимы трудные варианты 3SAT, ориентированные не на случайные входы, а на структурированные.

Основной целью данной статьи является описание простых преобразований для регулирования связности булевых функций, которые можно использовать как эффективный инструмент при генерации булевых функций и маскировке данных. На основе этих преобразований мы получим ряд трудных вариантов проблемы выполнимости, которые можно использовать не только для доказательства **NP**-трудности, но и для исследований за пределами **NP**.

1 Дизъюнктивно-связные булевы функции

Будем говорить, что литерал x принадлежит дизъюнкции C_i вида (2) и писать $x \in C_i$, если существует значение j такое, что $x = u_{i,j}$. Для любого литерала x будем полагать, что $x = \neg\neg x$. Для булевой функции f вида (1)рассмотрим граф $G_f = (V_f, E_f)$, заданный множеством вершин

$$V_f = \{C_i \mid 1 \leq i \leq m\} \tag{3}$$

и множеством ребер

$$E_f = \{(C_i, C_j) \mid 1 \leq i \leq m, 1 \leq j \leq m, \\ \exists x((x \in C_i \vee \neg x \in C_i) \wedge (x \in C_j \vee \neg x \in C_j))\}. \tag{4}$$

Следуя [53], булеву функцию f вида (1) будем называть дизъюнктивно-связной, если граф G_f является связным. Рассмотрим граф $D_f = (V_f, P_f)$, заданный множеством вершин (3) и множеством ребер

$$P_f = \{(C_i, C_j) \mid 1 \leq i \leq m, 1 \leq j \leq m, \\ \exists x(x \in C_i \wedge x \in C_j)\}. \tag{5}$$

Булеву функцию f вида (1) будем называть сильно дизъюнктивно-связной, если граф D_f является связным. Исходя из ограничений (4) и (5), накладываемых на ребра графа, у дизъюнктивно-связной булевой функции между дизъюнкциями существует связь тогда и только тогда, когда у них есть общая переменная, а у сильно дизъюнктивно-связной булевой функции между дизъюнкциями существует связь тогда и только тогда, когда у них есть общий литерал.

2 Дизъюнктивно-связная 3SAT

Рассмотрим сначала формулировку основной проблемы выполнимости в удобной для нас форме.

Проблема выполнимости (SAT).

Дано: *Булева функция* $f(x_1, x_2, \ldots, x_n) = \wedge_{i=1}^{m} C_i$, *где для любого* i, $1 \leq i \leq m$, *функция* C_i *является дизъюнкцией литералов.*

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

Согласно теореме Кука, проблема SAT является **NP**-полной. Хотя этого утверждения в явном виде в работе [54] нет, из рессуждений, представленных в статье [54], очевидным образом вытекает его справедливость.

Теперь рассмотрим проблему 3SAT, предложенную в книге [55]. Мы приведем формулировку проблемы 3SAT в удобной для нас форме, отличающейся от формулировки из [55] лишь терминологически.

Проблема 3-выполнимости (3SAT).

Дано: *Булева функция* $f(x_1, x_2, \ldots, x_n) = \wedge_{i=1}^{m} C_i$, *где для любого* i, $1 \leq i \leq m$, *функция* C_i *является дизъюнкцией ровно трех литералов.*

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

В книге [55] проблема 3SAT представлена как одна из **NP**-полных проблем из списка Карпа [56]. В книге [55] приведена полиномиальная сводимость проблемы SAT к 3SAT. Заметим, что в статье Карпа [56] в списке **NP**-полных проблем проблемы 3SAT нет. Карп рассматривал следующую проблему.

Проблема $\leq$3-выполнимости ($\leq$3SAT).

Дано: *Булева функция* $f(x_1, x_2, \ldots, x_n) = \wedge_{i=1}^{m} C_i$, *где для любого* i, $1 \leq i \leq m$, *функция* C_i *является дизъюнкцией не более трех литералов.*

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

Для каждой из трех рассмотренных формулировок проблемы выполнимости рассмотрим аналог, требующий, чтобы булева функция была дизъюнктивно-связной.

Проблема выполнимости дизъюнктивно-связной булевой функции (C-SAT).

Дано: *Булева функция* $f(x_1, x_2, \ldots, x_n) = \wedge_{i=1}^{m} C_i$ *такая, что граф* G_f *является связным и для любого* i, $1 \leq i \leq m$, *функция* C_i *является дизъюнкцией литералов.*

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

Проблема 3-выполнимости дизъюнктивно-связной булевой функции (C-3SAT).

Дано: *Булева функция* $f(x_1, x_2, \ldots, x_n) = \wedge_{i=1}^{m} C_i$ *такая, что граф* G_f *является связным и для любого* i, $1 \leq i \leq m$, *функция* C_i *является дизъюнкцией ровно трех литералов.*

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

Проблема $\leq$3-выполнимости дизъюнктивно-связной булевой функции (C-$\leq$3SAT).

Дано: *Булева функция* $f(x_1, x_2, \ldots, x_n) = \wedge_{i=1}^{m} C_i$ *такая, что граф* G_f *является связным и для любого* i, $1 \leq i \leq m$, *функция* C_i *является дизъюнкцией не более трех литералов.*

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

Заметим, что если булева функция $f(x_1, x_2, \ldots, x_n)$ не является дизъюнктивно-связной, то найдутся булевы функции

$$g(x_{i_1}, x_{i_2}, \ldots, x_{i_p}) = C_{j_1} \wedge C_{j_2} \wedge \cdots \wedge C_{j_q},$$

$$h(x_{r_1}, x_{r_2}, \ldots, x_{r_s}) = C_{l_1} \wedge C_{l_2} \wedge \cdots \wedge C_{l_t}$$

такие, что

$$n = p + s,$$

$$m = q + t,$$

$$\{x_1, x_2, \ldots, x_n\} = \{x_{i_1}, x_{i_2}, \ldots, x_{i_p}\} \cup \{x_{r_1}, x_{r_2}, \ldots, x_{r_s}\},$$
$$\{x_{i_1}, x_{i_2}, \ldots, x_{i_p}\} \cap \{x_{r_1}, x_{r_2}, \ldots, x_{r_s}\} = \emptyset,$$
$$\{C_1, C_2, \ldots, C_m\} = \{C_{j_1}, C_{j_2}, \ldots, C_{j_q}\} \cup \{C_{l_1}, C_{l_2}, \ldots, C_{l_t}\},$$
$$\{C_{j_1}, C_{j_2}, \ldots, C_{j_q}\} \cap \{C_{l_1}, C_{l_2}, \ldots, C_{l_t}\} = \emptyset.$$

В этом случае функция f будет выполняться тогда и только тогда, когда быдет выполнимой каждая из функций g и h. Соответственно, если у нас есть полиномиальный алгоритм для проверки выполнимости дизъюнктивно-связной булевой функции, то найдется и полиномиальный алгоритм для проверки выполнимости произвольной булевой функции, т.е. если проблема C-3SAT принадлежит классу **P**, то и проблема 3SAT принадлежит классу **P**. Однако, если **P** $\neq$ **NP** из только что рассмотренных рассуждений не следует автоматически, что 3SAT полиномиально сводится к C-3SAT, поскольку проверка выполнимости функции f по функциям g и h потребует **NP**-оракула. При решении задачи на практике функции **NP**-оракула может выполнить программа-решатель. Однако это не подходит для задач, требующих совместного решения нескольких проблем, и задач, в которых важна семантическая нагрузка булевой функции.

Для каждой из трех сформулированных проблем доказательство **NP**-полноты можно получить как следствие **NP**-полноты проблемы 3SAT, используя локальное преобразование, предложенное в работе [53]. Очевидно, что если у каждой пары дизъюнкций есть общая переменная, то граф G_f является связным. Если граф G_f не является связным, то найдутся дизъюнкции литералов $\vee_{i=1}^{p} u_i$ и $\vee_{j=1}^{q} v_j$, не имеющие общих переменных. Возьмем новую переменную y и заменим в функции $f(x_1, x_2, \ldots, x_n)$ подформулу

$$(\vee_{i=1}^{p} u_i) \wedge (\vee_{j=1}^{q} v_j) \tag{6}$$

на формулу

$$(\vee_{i=1}^{p} u_i) \wedge (\vee_{j=1}^{q} v_j) \wedge (u_p \vee v_q \vee y). \tag{7}$$

Получим функцию $g(x_1, x_2, \ldots, x_n, y)$. Очевидно, что замена формулы (6) на формулу (7) уменьшает количество пар дизъюнкций, у которых нет общих переменных. Соответственно функция g содержит на одну пару меньше, чем функция f. При этом поскольку все дизъюнкции функции f содержатся в функции g, из выполнимости функции g следует выполнимость функции f. Если функция f истинна на некотором наборе $t_1, t_2, \ldots, t_n$, то $g(t_1, t_2, \ldots, t_n, 1) = 1$. Следовательно, замена формулы (6) на формулу (7) сохраняет выполнимость. Легко понять, что последовательное применение замены формулы (6) на формулу (7) позволяет получить полиномиальную сводимость от проблемы 3SAT к проблеме C-3SAT. Учитывая очевидную принадлежность проблемы C-3SAT классу **NP**, это доказывает **NP**-полноту проблемы C-3SAT. Поскольку проблема C-3SAT является частным случаем проблем C-SAT и C-$\leq$3SAT, каждая из них тоже является **NP**-полной.

Заметим, что, осуществляя последовательные замены формул вида (6) на формулы вида (7), мы можем использовать одну и ту же переменную y. Поэтому мы можем получить итоговую функцию, содержащую единственную дополнительную переменную y, отсутствующую в начальном списке переменных.

Хотя использование замены формул вида (6) на формулы вида (7) обеспечивает простой способ перехода от произвольной булевой функции к дизъюнктивно-связной, у этого способа есть серьезный недостаток. При таком построении дизъюнктивно-связной булевой функции не наследуется количество решений, поскольку при определенных условиях мы, кроме решения вида $t_1, t_2, \ldots, t_n, 1$, можем получить решение вида $t_1, t_2, \ldots, t_n, 0$. Причем не всегда есть очевидный способ для отслеживания, следует ли из того, что $t_1, t_2, \ldots, t_n, 1$ является решением, то, что $t_1, t_2, \ldots, t_n, 0$ тоже будет решением. Заметим, что в общем случае проблема, требующая выяснить по данной булевой функции $f(x_1, x_2, \ldots, x_n)$ и данному решению $t_1, t_2, \ldots, t_n$, существует ли еще один набор, на котором функция f выполняется, является **NP**-полной [57].

Отсутствие сохранения количества решений при сводимости существенно затрудняет дальнейшее использование проблемы для исследований, связанных с многими важными классами, находящимися за пределами класса **NP**, что значительно уменьшает полезность преобразования на основе замены формул вида (6) на формулы вида (7) с теоретической точки зрения. С практической точки зрения преобразования, которые не сохраняют количество решений, тоже не всегда допустимы. В частности, их нельзя использовать совместно с #SAT-решателями [58, 59]. Во многих случаях недопустимо применять такие преобразования в криптографических целях. Их нельзя применять при проектировании квантовых вычислительных схем. Кроме того, исследования показывают, что при генерации случайных булевых функций для программ-решателей важно учитывать количество решений для этих функций даже в тех случаях, когда перед решателем ставится задача проверки выполнимости, а не нахождения решений [60].

3 Классы количественных алгоритмических проблем

На сегодняшний день предложено довольно большое количество сводимостей, которые можно использовать для исследования алгоритмических зависимостей между проблемами [61, 62, 63]. Для проблем из класса **NP** стандартной является полиномиальная сводимость.

Для произвольного множества Σ обозначим через Σ^* множество всевозможных слов над алфавитом Σ. Для произвольных языков L_1, $L_1 \subseteq \Sigma_1^*$, и L_2, $L_2 \subseteq \Sigma_2^*$, следуя [55], будем говорить, что язык L_1 полиномиально сводится к языку L_2, если существует детерминированная машина Тьюринга, вычисляющая за полиномиальное время функцию $f : \Sigma_1^* \to \Sigma_2^*$ такую, что $x \in L_1$ тогда и только тогда, когда $f(x) \in L_2$

для любого $x \in \Sigma_1^*$. Если язык L_1 полиномиально сводится к языку L_2 будем писать $L_1 \leq_T L_2$.

Язык L называется **NP**-полным, если $L \in \mathbf{NP}$ и $M \leq_T L$ для любого языка M из **NP**. Алгоритмическая проблема, предполагающая решение из множества $\{0,1\}$, называется **NP**-полной, если **NP**-полным является описывающий ее язык. Заметим, что если решение проблемы не обязательно из множества $\{0,1\}$, то проблема относится к классу функциональных проблем. В частности, через **FP** мы будем обозначать класс функциональных проблем, разрешимых детерминированной машиной Тьюринга за полиномиальное время. Заметим, что машина Тьюринга, устанавливающая полиномиальную сводимость языка L_1 к языку L_2, решает некоторую проблему f из класса **FP**.

Следуя [64], будем говорить, что язык L_1 логарифмически сводится к языку L_2, если существует детерминированная машина Тьюринга, вычисляющая функцию $f : \Sigma_1^* \to \Sigma_2^*$ с использованием памяти $O(\log n)$ такую, что $x \in L_1$ тогда и только тогда, когда $f(x) \in L_2$ для любого $x \in \Sigma_1^*$. Если язык L_1 логарифмически сводится к языку L_2 будем писать $L_1 \leq_L L_2$. Заметим, что если машина Тьюринга обеспечивает логарифмическую сводимость языка L_1 к языку L_2, то она всегда останавливается через полиномиальное количество шагов (см. [64], предложение 8.1). Поэтому из $L_1 \leq_L L_2$ следует $L_1 \leq_T L_2$. Таким образом, требование логарифмической сводимости, вообще говоря, более сильное, чем необходимо для доказательства **NP**-трудности. Однако логарифмическая сводимость в отличие от полиномиальной позволяет работать с подклассами класса **P** языков, распознаваемых детерминированной машиной Тьюринга за полиномиальное время. В частности, логарифмическую сводимость можно использовать для класса **NL** языков, распознаваемых недетерминированной машиной Тьюринга на логарифмической памяти. Кроме того, использование логарифмической сводимости представляет существенный практический интерес. В последние годы все большее внимание уделяется таким вопросам как автоматическая генерация тестов [65]; разработка языков программирования для решения задач при помощи решателей [66, 67, 68]; аппаратная реализация решателей [69, 70]; аппаратная реализация решателей как многоцелевых процессоров [71, 72]; аппаратная реализация программ, созданных решателями [73]. Все это указывает на постепенно формирующуюся тенденцию перехода от программного решения прикладных задач при помощи решателей к аппаратному, что требует экономного отношения к памяти при построении сводимостей. В дальнейшем под полиномиальной сводимостью языка L_1 к языку L_2 мы будем понимать логарифмическую и писать $L_1 \leq L_2$.

Рассмотрим теперь ряд важных классов, находящихся за пределами **NP**. Для произвольной алгоритмической проблемы $\mathcal{A}$ через $\#\mathcal{A}$ будем обозначать проблему, требующую по данному входу проблемы $\mathcal{A}$ найти

количество решений проблемы $\mathcal{A}$ на этом входе. Для произвольных исходных данных x и произвольной недетерминированной машины Тьюринга T через $\#acc_T(x)$ и $\#rej_T(x)$ мы будем обозначать количество допускающих и отвергающих вход x вычислений, соответственно. Через **GapP** мы будем обозначать класс всех функций f, для которых существует недетерминированная машина Тьюринга T такая, что равенство $f(x) = \#acc_T(x) - \#rej_T(x)$ выполняется для всех x. Следуя [74, 75], будем полагать, что

- **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 $x \in \Sigma^*$ [76, 77];
- $\mathbf{Mod}_k\mathbf{P}$ — класс языков $L \subseteq \Sigma^*$, для которых существует функция $f \in \mathbf{GapP}$ такая, что
$$x \in L \Leftrightarrow f(x) \not\equiv 0 \mod k$$
для любого $x \in \Sigma^*$ [78, 79, 80];
- $\oplus\mathbf{P} = \mathbf{Mod}_k\mathbf{P}$ [81, 82];
- **SPP** — класс языков $L \subseteq \Sigma^*$, для которых существует функция $f \in \mathbf{GapP}$ такая, что
$$x \in L \Rightarrow f(x) = 1,$$
$$x \notin L \Rightarrow f(x) = 0$$
для любого $x \in \Sigma^*$ [83];
- $\mathbf{C}_=\mathbf{P}$ — класс языков $L \subseteq \Sigma^*$, для которых существует функция $f \in \mathbf{GapP}$ такая, что
$$x \in L \Leftrightarrow f(x) = 0$$
для любого $x \in \Sigma^*$ [77, 84];
- **WPP** — класс языков $L \subseteq \Sigma^*$, для которых существуют функции $f \in \mathbf{GapP}$ и $g \in \mathbf{FP}$ такие, что для любого $x \in \Sigma^*$ выполняются соотношения
$$x \in L \Rightarrow f(x) = g(x),$$
$$x \notin L \Rightarrow f(x) = 0$$
и значение функции $g(x)$ отлично от нуля [83].

Заметим, что для этих классов известны следующие соотношения [74, 85]:

$$\mathbf{P} \subseteq \mathbf{SPP} \subseteq \mathbf{WPP} \subseteq \mathbf{C}_=\mathbf{P} \subseteq \mathbf{PP},$$
$$\mathbf{SPP} \subseteq \oplus\mathbf{P} \subseteq \mathbf{Mod}_k\mathbf{P},$$
$$\mathbf{co-NP} \subseteq \mathbf{C}_=\mathbf{P},$$
$$\mathbf{NP} \subseteq \mathbf{co-C}_=\mathbf{P},$$
$$\mathbf{WPP} \subseteq \mathbf{co-C}_=\mathbf{P},$$

где класс **co-NP** является дополнением класса **NP**, а **co-C$_=$P** обозначает дополнение класса **C$_=$P**.

Следуя [86], подсчитывающей машиной Тьюринга мы будем называть недетерминированную машину Тьюринга T с дополнительной лентой, предназначенной для печати в двоичном виде значения функции $\#acc_T(x)$. Обозначим через $t(n)$ максимальное время допустимого вычисления на входах, размер которых не превосходит n. Будем предполагать, что подсчитывающая машина Тьюринга в худшем случае имеет сложность по времени $t(n)$. Соответственно, трудоемкость генерации значения функции $\#acc_T(x)$ на дополнительной ленте не учитывается. Обозначим через $\#\mathbf{P}$ класс, состоящий из всех функций, которые могут быть вычислены подсчитывающей машиной Тьюринга за полиномиальное время [86].

Как и для большинства других классов вычислительной сложности, доказательство $\#\mathbf{P}$-трудности некоторой алгоритмической проблемы A обычно заключается в сведении к ней некоторой $\#\mathbf{P}$-полной проблемы B. Однако сводимость должна обеспечивать сохранение количества решений. Для произвольного входа I алгоритмической проблемы Π обозначим через $\#I$ количество решений проблемы Π на входе I.

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 $\#I_1 = \#I_2$, то полиномиальная сводимость R проблемы Π_1 к проблеме Π_2 называется экономной [87]. Для экономной сводимости мы будем использовать обозначение $\Pi_1 \leq_E \Pi_2$.

По произвольной функции $\#\mathcal{A} : \Sigma^* \to \mathbb{N}$ из класса $\#\mathbf{P}$, следуя [88], определим следующие алгоритмические проблемы.

- $\oplus\mathcal{A}$ — по произвольному входу $x \in \Sigma^*$ выяснить, является ли значение $\#\mathcal{A}(x)$ нечетным.
- $\mathbf{Mod}_k\mathcal{A}$ — по произвольному входу $x \in \Sigma^*$ выяснить, верно ли, что $\#\mathcal{A}(x) \not\equiv 0 \mod k$.
- $\mathbf{Diff}\mathcal{A}_{=0}$ — по произвольному входу $(x, y) \in \Sigma^* \times \Sigma^*$ выяснить, верно ли, что $\#\mathcal{A}(x) = \#\mathcal{A}(y)$.
- $\mathbf{Diff}\mathcal{A}_{>0}$ — по произвольному входу $(x, y) \in \Sigma^* \times \Sigma^*$ выяснить, верно ли, что $\#\mathcal{A}(x) > \#\mathcal{A}(y)$.
- $\mathbf{Diff}\mathcal{A}_{=1}$ — по произвольному входу $(x, y) \in \Sigma^* \times \Sigma^*$ в предположении, что

$$(x, y) \in I_{yes} \cup I_{no},$$

где

$$I_{yes} = \{(x, y) \mid \#\mathcal{A}(x) = \#\mathcal{A}(y) + 1\},$$

$$I_{no} = \{(x, y) \mid \#\mathcal{A}(x) = \#\mathcal{A}(y)\},$$

выяснить, верно ли, что

$$(x, y) \in I_{yes}.$$

- **Diff**$\mathcal{A}_{=g}$ — по произвольному входу $(x, y, k) \in \Sigma^* \times \Sigma^* \times \mathbb{N}$ в предположении, что

$$(x, y, k) \in I_{yes} \cup I_{no},$$

где

$$I_{yes} = \{(x, y, k) \mid \#\mathcal{A}(x) = \#\mathcal{A}(y) + k\},$$
$$I_{no} = \{(x, y, k) \mid \#\mathcal{A}(x) = \#\mathcal{A}(y)\},$$

выяснить, верно ли, что

$$(x, y, k) \in I_{yes}.$$

Согласно [88], если проблема $\#\mathcal{A}$ является $\#$**P**-полной относительно экономной сводимости, то проблемы $\oplus\mathcal{A}$, **Mod**$_k\mathcal{A}$, **Diff**$\mathcal{A}_{=0}$, **Diff**$\mathcal{A}_{>0}$, **Diff**$\mathcal{A}_{=1}$, **Diff**$\mathcal{A}_{=g}$ являются полными для классов $\oplus$**P**, **Mod**$_k$**P**, **C**$_=$**P**, **PP**, **SPP**, **WPP**, соответственно.

4 Экономная сводимость для 3SAT

Доказательство теоремы Кука [54], которое устанавливает **NP**-трудность проблемы SAT, является первым примером непосредственной интерпретации машины Тьюринга при доказательстве **NP**-полноты алгоритмической проблемы. Такой подход используется весьма редко. Для большинства алгоритмических проблем доказательство трудности осуществляется демонстрацией полиномиальной сводимости от SAT или от проблемы, для которой сводимость от SAT уже известна.

Следует отметить, что в работе [54] булева функция $f(x_1, x_2, \ldots, x_n)$ строится так, что каждому допустимому вычислению машины Тьюринга соответствует ровно один набор значений переменных, для которого $f(x_1, x_2, \ldots, x_m) = 1$. Следуя рассуждениям Кука, представленным в работе [54], несложно убедиться в том, что проблема #SAT является $\#$**P**-полной относительно экономной сводимости.

Легко понять, что в книге [55] представлена формулировка проблемы 3-выполнимости, которая более удобна для использования при доказательстве трудности различных проблем. Проблема $\leq$3SAT предполагает, что в функции $f(x_1, x_2, \ldots, x_n)$ нет дизъюнкций более трех литералов, а у проблемы 3SAT нет и дизъюнкций более трех литералов, и дизъюнкций менее трех литералов. Однако, полиномиальная сводимость, построенная в [56], является экономной, а полиномиальная сводимость, предложенная в [55], экономной не является. В самом деле, доказательство в [56] основано на замене произвольной дизъюнкции литералов

$$\sigma_1 \vee \sigma_2 \vee \cdots \vee \sigma_m, \tag{8}$$

где $m > 3$, формулой

$$(\sigma_1 \vee \sigma_2 \vee u_1) \wedge (\sigma_3 \vee \sigma_4 \vee \cdots \vee \sigma_m \vee \neg u_1) \wedge$$
$$(\neg\sigma_3 \vee u_1) \wedge (\neg\sigma_4 \vee u_1) \wedge \cdots \wedge (\neg\sigma_m \vee u_1), \tag{9}$$

где u_1 является новой переменной. Очевидно, что формула (9) выполняется тогда и только тогда, когда выполняются все ее конъюнкты. Если для некоторого $3 \leq i \leq m$ имеет место равенство $\sigma_i = 1$, то для выполнимости

$$(\neg\sigma_3 \vee u_1) \wedge (\neg\sigma_4 \vee u_1) \wedge \cdots \wedge (\neg\sigma_m \vee u_1), \tag{10}$$

необходимо, чтобы $u_1 = 1$. Если для любого $3 \leq i \leq m$ имеет место равенство $\sigma_i = 0$, то для выполнимости

$$\sigma_3 \vee \sigma_4 \vee \cdots \vee \sigma_m \vee \neg u_1, \tag{11}$$

необходимо, чтобы $u_1 = 0$. Таким образом, формулы (10) и (11) обеспечивают единственность решения (9) для каждого решения (8). Несложно убедиться в том, что отсюда вытекает, что сводимость, которую в работе [56] предложил Карп, является экономной сводимостью от проблемы SAT к проблеме $\leq$3SAT. Соответственно, проблема $\#\leq$3SAT является $\#\mathbf{P}$-полной относительно экономной сводимости. С другой стороны, замена дизъюнкции литералов

$$\sigma_1 \vee \sigma_2 \tag{12}$$

формулой

$$(\sigma_1 \vee \sigma_2 \vee u_1) \wedge (\sigma_1 \vee \sigma_2 \vee \neg u_1), \tag{13}$$

предлагаемая в [55], вообще говоря, позволяет для каждого решения (12) получить два решения (13). Поэтому сводимость, предложенная в книге [55], экономной не является. Во многих случаях это делает сводимость, предложенную в книге [55], неудобной для дальнейшего использования. Учитывая то, что проблема 3SAT является наиболее популярной при доказательстве трудности, естественно стремиться получить для нее не только наиболее эффективную сводимость, но и максимально выразительную формулировку, поскольку во многих случаях свойства, которые могли бы быть сформулированы для проблемы 3SAT, приходится многократно воспроизводить при построении сводимостей (см., например, [89, 90]).

В формулировке $\leq$3SAT можно предполагать, что в одной и той же дизъюнкции литерал можно использовать, вообще говоря, несколько раз. Тогда мы легко получим 3SAT и сохраним экономность сводимости. Например, дизъюнкции u и $u \vee v$ можно заменять на дизъюнкции $u \vee u \vee u$ и $u \vee v \vee v$, соответственно. Однако допущение многократного использования литералов при доказательстве трудности проблем, например, для графов может привести к появлению кратных вершин и ребер. Разбор соответствующих случаев может сделать доказательства существенно более громоздкими. Аналогичные проблемы возникнут и для многих других прикладных задач.

Несложной модификацией доказательства, предложенного в [55], можно сохранить экономность сводимости без использования повторяющихся литералов. В частности, вместо локальной замены, предложенной в

[55], можно использовать локальную замену следующего вида. Произвольная дизъюнкция σ_1 заменяется на формулу

$$\begin{gathered}(\sigma_1 \vee u_1 \vee u_2) \wedge (\neg u_1 \vee \neg u_2 \vee \neg u_3) \wedge \\ (u_1 \vee \neg u_2 \vee \neg u_3) \wedge (\neg u_1 \vee u_2 \vee \neg u_3) \wedge \\ (\neg u_1 \vee \neg u_2 \vee u_3) \wedge (u_1 \vee u_2 \vee \neg u_3) \wedge \\ (u_1 \vee \neg u_2 \vee u_3) \wedge (\neg u_1 \vee u_2 \vee u_3),\end{gathered} \tag{14}$$

дизъюнкция $\sigma_1 \vee \sigma_2$ заменяется на формулу

$$\begin{gathered}(\sigma_1 \vee \sigma_2 \vee u_1) \wedge (\neg u_1 \vee \neg u_2 \vee \neg u_3) \wedge \\ (u_1 \vee \neg u_2 \vee \neg u_3) \wedge (\neg u_1 \vee u_2 \vee \neg u_3) \wedge \\ (\neg u_1 \vee \neg u_2 \vee u_3) \wedge (u_1 \vee u_2 \vee \neg u_3) \wedge \\ (u_1 \vee \neg u_2 \vee u_3) \wedge (\neg u_1 \vee u_2 \vee u_3),\end{gathered} \tag{15}$$

дизъюнкция (8) заменяется на формулу

$$\begin{gathered}(\sigma_1 \vee \sigma_2 \vee u_1) \wedge (\sigma_3 \vee \sigma_4 \vee \cdots \vee \sigma_m \vee \neg u_1) \wedge \\ \wedge_{i \neq j, m \geq i > 2, m \geq j > 2} (\neg \sigma_i \vee \neg \sigma_j \vee u_1) \wedge \\ \wedge_{i \neq j, m \geq i > 2, m \geq j > 2} (\neg \sigma_i \vee \sigma_j \vee u_1).\end{gathered} \tag{16}$$

Заметим, что формула

$$\begin{gathered}(\neg u_1 \vee \neg u_2 \vee \neg u_3) \wedge (u_1 \vee \neg u_2 \vee \neg u_3) \wedge \\ (\neg u_1 \vee u_2 \vee \neg u_3) \wedge (\neg u_1 \vee \neg u_2 \vee u_3) \wedge \\ (u_1 \vee u_2 \vee \neg u_3) \wedge (u_1 \vee \neg u_2 \vee u_3) \wedge \\ (\neg u_1 \vee u_2 \vee u_3)\end{gathered} \tag{17}$$

выполняется тогда и только тогда, когда

$$u_1 = u_2 = u_3 = 0. \tag{18}$$

Выполнимость формулы (14) или (15) требует выполнимости формулы (17). Поэтому выполнимость формулы (14) равносильна выполнимости дизъюнкции σ_1 и системы равенств (18). Аналогично выполнимость формулы (15) равносильна выполнимости дизъюнкции $\sigma_1 \vee \sigma_2$ и системы равенств (18). Поэтому замена дизъюнкций σ_1 и $\sigma_1 \vee \sigma_2$ на формулы (14) и (15), соответственно, не влияет на количество решений. Более того, легко понять, что для всех дизъюнкций вида σ_1 и $\sigma_1 \vee \sigma_2$ мы можем использовать единую тройку переменных u_1, u_2 и u_3. Выполнимость формулы (16) требует выполнимости формул (11) и

$$\begin{gathered}\wedge_{i \neq j, m \geq i > 2, m \geq j > 2} (\neg \sigma_i \vee \neg \sigma_j \vee u_1) \wedge \\ \wedge_{i \neq j, m \geq i > 2, m \geq j > 2} (\neg \sigma_i \vee \sigma_j \vee u_1).\end{gathered} \tag{19}$$

Если для некоторого $3 \leq i \leq m$ имеет место равенство $\sigma_i = 1$, то для выполнимости формулы (19) необходимо, чтобы $u_1 = 1$. Если для любого $3 \leq i \leq m$ имеет место равенство $\sigma_i = 0$, то для выполнимости формулы

(11) необходимо, чтобы $u_1 = 0$. Таким образом, формулы (11) и (19) обеспечивают единственность решения (16) для каждого решения (8).

Таким образом, мы получили экономную сводимость от проблемы SAT к проблеме 3SAT. Соответственно, проблема #3SAT является #**P**-полной относительно экономной сводимости. При этом мы можем предполагать, что в каждой дизъюнкции три литерала, соответствующие трем различным переменным.

5 Экономная сводимость для C-3SAT

Теорема 1. 3SAT $\leq_E$ C-3SAT.

Доказательство. Если на входе у проблемы 3SAT дизъюнктивно-связная булева функция, то ничего делать не нужно. Поэтому без ограничения общности можно предполагать, что булева функция на входе у проблемы 3SAT дизъюнктивно-связной не является.

Рассмотрим произвольную булеву функцию $f(x_1, x_2, \ldots, x_n)$ вида (1), где для любого i, $1 \leq i \leq m$, функция C_i имеет вид (2) и является дизъюнкцией ровно трех литералов. Обозначим через X множество всевозможных пар дизъюнкций (C_i, C_j), $1 \leq i \leq m$, $1 \leq j \leq m$, не имеющих общих переменных. В функции $f(x_1, x_2, \ldots, x_n)$ произвольную формулу

$$C_i \wedge C_j, \tag{20}$$

где $(C_i, C_j) \in X$, заменим на формулу

$$\begin{gathered} C_i \wedge C_j \wedge \\ (y_1 \vee u_{i,3} \vee u_{j,3}) \wedge \\ (y_1 \vee y_2 \vee y_3) \wedge (y_1 \vee \neg y_2 \vee \neg y_3) \wedge \\ (\neg y_1 \vee y_2 \vee \neg y_3) \wedge (\neg y_1 \vee \neg y_2 \vee y_3) \wedge \\ (y_1 \vee y_2 \vee \neg y_3) \wedge (y_1 \vee \neg y_2 \vee y_3) \wedge \\ (\neg y_1 \vee y_2 \vee y_3), \end{gathered} \tag{21}$$

где y_1, y_2, y_3 являются новыми переменными. Получим булеву функцию

$$\begin{gathered} g(x_1, x_2, \ldots, x_n, y_1, y_2, y_3) = \\ f(x_1, x_2, \ldots, x_n) \wedge \\ (y_1 \vee u_{i,3} \vee u_{j,3}) \wedge \\ (y_1 \vee y_2 \vee y_3) \wedge (y_1 \vee \neg y_2 \vee \neg y_3) \wedge \\ (\neg y_1 \vee y_2 \vee \neg y_3) \wedge (\neg y_1 \vee \neg y_2 \vee y_3) \wedge \\ (y_1 \vee y_2 \vee \neg y_3) \wedge (y_1 \vee \neg y_2 \vee y_3) \wedge \\ (\neg y_1 \vee y_2 \vee y_3). \end{gathered} \tag{22}$$

Предположим, что булева функция $f(x_1, x_2, \ldots, x_n)$ выполняется на наборе $t_1, t_2, \ldots, t_n$. Заметим, что формула

$$\begin{gathered}(y_1 \vee y_2 \vee y_3) \wedge (y_1 \vee \neg y_2 \vee \neg y_3) \wedge \\ (\neg y_1 \vee y_2 \vee \neg y_3) \wedge (\neg y_1 \vee \neg y_2 \vee y_3) \wedge \\ (y_1 \vee y_2 \vee \neg y_3) \wedge (y_1 \vee \neg y_2 \vee y_3) \wedge \\ (\neg y_1 \vee y_2 \vee y_3)\end{gathered} \tag{23}$$

выполняется тогда и только тогда, когда

$$y_1 = y_2 = y_3 = 1. \tag{24}$$

Поэтому выполняется равенство

$$g(t_1, t_2, \ldots, t_n, 1, 1, 1) = 1$$

и для любых a_1, a_2, a_3 таких, что

$$a_1 \wedge a_2 \wedge a_3 = 0,$$

имеет место равенство

$$g(t_1, t_2, \ldots, t_n, a_1, a_2, a_3) = 0.$$

Следовательно, из выполнимости функции f вытекает выполнимость функции g. При этом каждому набору значений переменных $t_1, t_2, \ldots, t_n$, на котором выполняется функция f, соответствует единственный набор значений переменных $t_1, t_2, \ldots, t_n, 1, 1, 1$, на котором выполняется функция g.

Допустим теперь, что функция g выполняется на некотором наборе значений переменных

$$t_1, t_2, \ldots, t_{n+3} \tag{25}$$

Поскольку формула (23) выполняется только при условии (24), набор (25) должен иметь вид

$$t_1, t_2, \ldots, t_n, 1, 1, 1. \tag{26}$$

Подставив набор (26) в соотношение (22), получим

$$g(t_1, t_2, \ldots, t_n, 1, 1, 1) = f(t_1, t_2, \ldots, t_n).$$

Следовательно, функция f выполняется на наборе $t_1, t_2, \ldots, t_n$.

Таким образом, замена формулы (20) на формулу (21) сохраняет выполнимость и количество решений. Несложно убедиться в том, что, осуществив замену формулы вида (20) на формулу вида (21) для каждой пары $(C_i, C_j) \in X$, мы сохраним выполнимость и количество решений. Более того, легко понять, что для всех пар $(C_i, C_j) \in X$ мы можем использовать одни и те же переменные y_1, y_2, y_3.

Рассмотрим граф $G_g = (V_g, E_g)$. По определению

$$\begin{gathered}V_g = V_f \cup \{(y_1 \vee u_{i,3} \vee u_{j,3}), (y_1 \vee y_2 \vee y_3), \\ (y_1 \vee \neg y_2 \vee \neg y_3), (\neg y_1 \vee y_2 \vee \neg y_3), \\ (\neg y_1 \vee \neg y_2 \vee y_3), (y_1 \vee y_2 \vee \neg y_3),\end{gathered}$$

$$(y_1 \vee \neg y_2 \vee y_3), (\neg y_1 \vee y_2 \vee y_3)\}.$$

Соответственно, каждая замена формулы (20) на формулу (21) добавляет новые вершины, которые смежны между собой, поскольку каждая из новых дизъюнкций содержит переменную y_1. После замены формулы вида (20) на формулу вида (21) для каждой пары $(C_i, C_j) \in X$ все вершины множества V_f попадут в одну компоненту связности, поскольку для любой пары дизъюнкций (C_i, C_j), $1 \leq i \leq m$, $1 \leq j \leq m$, либо вершины C_i и C_j являются смежными, либо существует путь

$$(C_i, (y_1 \vee u_{i,3} \vee u_{j,3}), C_j).$$

Каждая группа новых вершин, соответствующая замене формулы вида (20) на формулу вида (21), образует полный подграф, все вершины которого принадлежат той же компоненте связности, что и V_f, благодаря ребрам $(C_i, (y_1 \vee u_{i,3} \vee u_{j,3}))$ и $((y_1 \vee u_{i,3} \vee u_{j,3}), C_j)$. Таким образом, замена формулы вида (20) на формулу вида (21) для каждой пары $(C_i, C_j) \in X$ позволяет получить дизъюнктивно-связную булеву функцию. Для завершения доказательства теоремы осталось заметить, что замена формулы вида (20) на формулу вида (21) добавляет в булеву функцию только дизъюнкции, состоящие ровно из трех литералов. □

Заметим, что проблема C-3SAT является частным случаем проблем C-≤3SAT и C-SAT. Поэтому из соотношения 3SAT $\leq_E$ C-3SAT следуют соотношения 3SAT $\leq_E$ C-SAT и 3SAT $\leq_E$ C-≤3SAT. Отсюда вытекает следующее утверждение.

Следствие 1. *Проблемы* #C-3SAT, #C-≤3SAT *и* #C-SAT *являются* #**P**-*полными относительно экономной сводимости.*

Рассмотрим теперь следующую проблему.

Проблема 3-выполнимости сильно дизъюнктивно-связной булевой функции (SC-3SAT).

Дано: *Булева функция* $f(x_1, x_2, \ldots, x_n) = \wedge_{i=1}^{m} C_i$ *такая, что граф* D_f *является связным и для любого* i, $1 \leq i \leq m$, *функция* C_i *является дизъюнкцией ровно трех литералов.*

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

Теорема 2. 3SAT $\leq_E$ SC-3SAT.

Доказательство. Если на входе у проблемы 3SAT сильно дизъюнктивно-связная булева функция, то ничего менять не требуется. Поэтому мы можем предполагать, что булева функция на входе у проблемы 3SAT не является сильно дизъюнктивно-связной.

Для произвольной булевой функции $f(x_1, x_2, \ldots, x_n)$ вида (1), где для любого i, $1 \leq i \leq m$, функция C_i имеет вид (2) и является дизъюнкцией ровно трех литералов, рассмотрим множество X всевозможных пар дизъюнкций (C_i, C_j), $1 \leq i \leq m$, $1 \leq j \leq m$, не имеющих общих литералов. При замене формулы вида (20) на формулу вида (21) из функции f

мы получаем функцию g вида (22). Легко понять, что граф D_g содержит пути

$$C_i, (y_1 \vee u_{i,3} \vee u_{j,3}), C_j),$$
$$((y_1 \vee u_{i,3} \vee u_{j,3}), (y_1 \vee y_2 \vee y_3),$$
$$(y_1 \vee \neg y_2 \vee \neg y_3), (\neg y_1 \vee y_2 \vee \neg y_3),$$
$$(\neg y_1 \vee \neg y_2 \vee y_3), (y_1 \vee \neg y_2 \vee y_3),$$
$$(\neg y_1 \vee y_2 \vee y_3), (y_1 \vee y_2 \vee \neg y_3)).$$

Поэтому для произвольной пары (C_i, C_j) из множества X замена формулы (20) на формулу (21) обеспечит сильную дизъюнктивную связность булевой функции. Однако замена формулы вида (20) на формулу вида (21) может привести к появлению дизъюнкции, содержащей литералы, соответствующие одной и той же переменной. Мы предполагаем, что в функции $f(x_1, x_2, \ldots, x_n)$ каждая дизъюнкция содержит литералы, соответствующие трем различным переменным. Поэтому для произвольной пары дизъюнкций (C_i, C_j) мы всегда можем выбрать литералы $u_{i,k}$ и $u_{j,p}$, соответствующие различным переменным. Соответственно, замена произвольной формулы (20) на формулу

$$\begin{gathered} C_i \wedge C_j \wedge \\ (y_1 \vee u_{i,k} \vee u_{j,p}) \wedge \\ (y_1 \vee y_2 \vee y_3) \wedge (y_1 \vee \neg y_2 \vee \neg y_3) \wedge \\ (\neg y_1 \vee y_2 \vee \neg y_3) \wedge (\neg y_1 \vee \neg y_2 \vee y_3) \wedge \\ (y_1 \vee y_2 \vee \neg y_3) \wedge (y_1 \vee \neg y_2 \vee y_3) \wedge \\ (\neg y_1 \vee y_2 \vee y_3) \end{gathered} \tag{27}$$

не приведет к появлению дизъюнкции, содержащей литералы, соответствующие одной и той же переменной. По аналогии с доказательством теоремы 1 мы можем убедиться в том, что использование замены формулы вида (20) на формулу вида (27) позволяет обеспечить требуемую сводимость. □

Непосредственно из теоремы 2 получаем следующее утверждение.

Следствие 2. *Проблема* #SC-3SAT *является* #**P**-*полной относительно экономной сводимости.*

Заметим, что при доказательстве теоремы 2 мы пользовались тем, что каждая дизъюнкция булевой функции проблемы 3SAT содержит литералы, соответствующие трем различным переменным, т.е. вырожденных дизъюнкций нет. Однако различные прикладные задачи могут допускать использование вырожденных дизъюнкций. В частности, это может возникать при случайной генерации булевых функций, маскировке криптографических данных, совместном решении нескольких задач. В некоторых случаях может потребоваться изменить плотность булевой функции, сохраняя имеющиеся вырожденные дизъюнкции, но не

добавляя новых. В частности, это может представлять интерес при разработке аппаратных вредоносных программ. В этом случае для некоторой переменной x пара дизъюнкций (C_i, C_j) может принять вид $(x, \neg x)$ (совместная выполнимость такой пары дизъюнкций обеспечивается через провокацию отказа системы и нарушение целостности данных). Для такой пары мы можем использовать замену на формулу

$$\begin{gathered} C_i \wedge C_j \wedge \\ (y_1 \vee y_2 \vee x) \wedge (y_1 \vee y_2 \vee \neg x) \wedge \\ (y_1 \vee y_2 \vee y_3) \wedge (y_1 \vee \neg y_2 \vee \neg y_3) \wedge \\ (\neg y_1 \vee y_2 \vee \neg y_3) \wedge (\neg y_1 \vee \neg y_2 \vee y_3) \wedge \\ (y_1 \vee y_2 \vee \neg y_3) \wedge (y_1 \vee \neg y_2 \vee y_3) \wedge \\ (\neg y_1 \vee y_2 \vee y_3). \end{gathered}$$

6 Планарная дизъюнктивно-связная выполнимость

В работе [89] было введено понятие планарной булевой функции и доказана **NP**-полнота планарной проблемы 3-выполнимости. В последующих исследованиях была доказана вычислительная трудность для многих планарных вариантов проблемы выполнимости [87, 91, 92]. Однако в этих работах использовался подход к построению графа, который отличается от рассматриваемого в данной работе. В этом разделе мы рассмотрим проблему выполнимости для планарных дизъюнктивно-связных булевых функций.

Проблема $\leq$3-выполнимости планарной дизъюнктивно-связной булевой функции (PC-$\leq$3SAT).

Дано: *Булева функция* $f(x_1, x_2, \ldots, x_n) = \wedge_{i=1}^{m} C_i$ *такая, что функция* C_i, $1 \leq i \leq m$, *является дизъюнкцией не более трех литералов и граф* G_f *является планарным и связным.*

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

Теорема 3. 3SAT $\leq_E$ PC-$\leq$3SAT.

Доказательство. Доказательство теоремы в существенной мере будет опираться на конструкцию, предложенную в работе [89], и результаты работы [92]. Рассмотрим булеву функцию $f(x_1, x_2, \ldots, x_n) = \wedge_{i=1}^{m} C_i$, где для любого i, $1 \leq i \leq m$, функция C_i является дизъюнкцией ровно трех литералов. Определим для функции f граф $G(f)$ в соответствие с подходом, предложенным в работе [89]. Пусть $G(f) = (V(f), E(f))$ — граф, заданный множеством вершин $V(f)$ и множеством ребер $E(f)$, где

$$\begin{gathered} V(f) = \{C_i \mid 1 \leq i \leq m\} \cup \{x_j \mid 1 \leq j \leq n\}, \\ E(f) = E_1 \cup E_2 \cup E_3 \cup E_4, \\ E_1 = \{(C_i, x_j) \mid 1 \leq i \leq m, 1 \leq j \leq n, x_j \in C_i\}, \\ E_2 = \{(C_i, x_j) \mid 1 \leq i \leq m, 1 \leq j \leq n, \neg x_j \in C_i\}, \end{gathered}$$

$$E_3 = \{(x_j, x_{j+1}) \mid 1 \le j \le n-1\},$$
$$E_4 = \{(x_n, x_1)\}.$$

Таким образом, все ребра графа $G(f)$, кроме простого цикла, образованного переменными, расположенными в естественном порядке, соединяют некоторую дизъюнкцию с некоторой переменной.

В работе [89] предложен подход на основе локальной замены к преобразованию булевой функции f в некоторую булеву функцию g, имеющую планарный граф $G(g)$. Первый этап построения сводимости 3SAT $\le_E$ PC-$\le$3SAT будет основан на этой конструкции.

Предполагается, что цикл, образованный переменными, изображается естественным образом. Вершины графа, соответствующие дизъюнкциям, располагаются с внешней стороны цикла в произвольном порядке. В частности, можно предполагать, что переменные и дизъюнкции располагаются на двух параллельных линиях. Мы будем предполагать, что дизъюнкции соединяются с переменными отрезками прямых линий. Легко понять, что мы можем предполагать, что пересекаться могут только ребра, каждое из которых смежно с некоторой дизъюнкцией. При этом если какие-либо ребра пересекаются, то всегда найдется пара ребер (C_i, x_s) и (C_j, x_t) такая, что ребра (C_i, x_s) и (C_j, x_t) пересекаются в точке $\mathcal{W}$ и отрезки $[\mathcal{W}, x_s]$ и $[\mathcal{W}, x_t]$ не имеют пересечений с ребрами. На отрезках $[C_i, \mathcal{W}]$ и $[C_j, \mathcal{W}]$ мы всегда можем расположить точки $\mathcal{U}$ и $\mathcal{V}$, соответственно, так, что отрезки $[\mathcal{U}, \mathcal{W}]$ и $[\mathcal{V}, \mathcal{W}]$ не имеют пересечений с ребрами. Точки $\mathcal{U}$, $\mathcal{V}$ и $\mathcal{W}$ сделаем вершинами нового графа, полагая

$$G = (V, E),$$
$$V = V(f) \cup \{\mathcal{U}, \mathcal{V}, \mathcal{W}\},$$
$$E = (E(f) \backslash \{(C_i, x_s), (C_j, x_t)\}) \cup$$
$$\{(C_i, \mathcal{U}), (C_j, \mathcal{V}), (\mathcal{U}, \mathcal{W}), (\mathcal{V}, \mathcal{W}), (\mathcal{W}, x_s), (\mathcal{W}, x_t)\}.$$

После этого в конструкции, предложенной в работе [89], подграф, порожденный множеством вершин

$$\{x_s, x_t, \mathcal{U}, \mathcal{V}, \mathcal{W}\},$$

заменяется на планарный граф специального вида, который смежен с оставшейся частью графа по вершинам

$$x_s, x_t, \mathcal{U}, \mathcal{V}.$$

Для того чтобы полученный граф сделать графом некоторой булевой функции, конструкция, предложенная в работе [89], предполагает ряд изменений в булевой функции f и ряд переобозначений в самом графе. Без ограничения общности можно предполагать, что $C_i = u_{i,1} \vee u_{i,2} \vee u_{i,3}$, где $u_{i,3} \in \{x_s, \neg x_s\}$, $C_j = u_{j,1} \vee u_{j,2} \vee u_{j,3}$, где $u_{j,3} \in \{x_t, \neg x_t\}$. Обозначим вершины $\mathcal{U}$ и $\mathcal{V}$ через y_s и y_t, соответственно. Если $C_i = u_{i,1} \vee u_{i,2} \vee x_s$, то заменим C_i на $C_i' = u_{i,1} \vee u_{i,2} \vee y_s$. Если $C_i = u_{i,1} \vee u_{i,2} \vee \neg x_s$, то заменим C_i на $C_i' = u_{i,1} \vee u_{i,2} \vee \neg y_s$. Если $C_j = u_{j,1} \vee u_{j,2} \vee x_t$, то заменим C_j на $C_j' = u_{j,1} \vee u_{j,2} \vee y_t$. Если $C_j = u_{j,1} \vee u_{j,2} \vee \neg x_t$, то заменим C_j на

$C'_j = u_{j,1} \vee u_{j,2} \vee \neg y_t$. Объявим вершины y_s и y_t новыми вершинами переменных. Кроме того, добавим в множество вершин переменных шесть внутренних вершин планарного подграфа: $u_1, u_2, v_1, v_2, v_3, v_4$. Модифицируем функцию f при помощи замены подформулы

$$C_i \wedge C_j \tag{28}$$

на подформулу

$$\begin{gathered} C'_i \wedge C'_j \wedge \\ (\neg u_1 \vee \neg u_2 \vee v_1) \wedge (u_1 \vee \neg v_1) \wedge (u_2 \vee \neg v_1) \wedge \\ (\neg u_1 \vee y_t \vee v_2) \wedge (u_1 \vee \neg v_2) \wedge (\neg y_t \vee \neg v_2) \wedge \\ (y_s \vee y_t \vee v_3) \wedge (\neg y_s \vee \neg v_3) \wedge (\neg y_t \vee \neg v_3) \wedge \\ (y_s \vee \neg u_2 \vee v_4) \wedge (\neg y_s \vee \neg v_4) \wedge (u_2 \vee \neg v_4) \wedge \\ (v_1 \vee v_2 \vee v_3 \vee v_4) \wedge \\ (\neg v_1 \vee \neg v_2) \wedge (\neg v_2 \vee \neg v_3) \wedge (\neg v_3 \vee \neg v_4) \wedge (\neg v_1 \vee \neg v_4) \wedge \\ (u_1 \vee \neg x_s) \wedge (\neg u_1 \vee x_s) \wedge (u_2 \vee \neg x_t) \wedge (\neg u_2 \vee x_t). \end{gathered} \tag{29}$$

Таким образом, замена формулы вида (28) на формулу вида (29) заключается в замене двух дизъюнкций C_i и C_j на их модифицированные варианты C'_i и C'_j и добавлении 21 новой дизъюнкции. При этом каждая из новых дизъюнкций соответствует одной из вершин планарного подграфа. На этом построение планарного графа и соответствующей ему булевой функции для выделенного пересечения ребер полностью завершается. В дальнейшем на первом этапе построения сводимости добавленный планарный подграф и вершины переменных x_s и x_t модифицироваться не будут. Изменения могут затронуть только вершины дизъюнкций C_i и C_j. Для завершения первого этапа построения сводимости нам необходимо последовательно перебрать все пересечения ребер, осуществляя для каждого из них только что описанные действия. Корректность этой процедуры обоснована в работе [89]. Следует отметить, что замена формулы вида (28) на формулу вида (29) позволяет получить из функции f некоторую функцию g такую, что функция f выполняется тогда и только тогда, когда выполняется функция g, причем замена формулы вида (28) на формулу вида (29) сохраняет количество решений [89].

До сих пор в тексте при доказательствах мы не упоминали логарифмическую память, поскольку выполнимость соответствующих алгоритмов на логарифмической памяти была тривиальной. Теперь же нам следует сделать ряд существенных замечаний, связанных с реализуемостью данной сводимости в логарифмической памяти.

- Как обычно, логарифмическая сводимость подразумевает, что у нас есть три области памяти. Одна из них изначально содержит исходные данные и предназначена только для чтения. Вторая изначально является пустой и предназначена только для записи

результата. Кроме того, есть область памяти, которая изначально является пустой и доступна как для чтения, так и для записи, но размер этой области логарифмически ограничен размером исходных данных. Поэтому суперпозиция логарифмических сводимостей, вообще говоря, логарифмической сводимостью не является. Сводимость 3SAT $\leq_E$ PC-$\leq$3SAT мы описываем как цепь последовательных преобразований. Формально эта последовательность преобразований логарифмической сводимостью не будет, поскольку по завершении каждого этапа преобразований формируется новый граф, размер которого сравним с размером исходного графа и не позволяет новому графу полностью разместиться в логарифмической памяти. Поэтому, несмотря на последовательное изложение преобразований, мы предполагаем, что все этапы преобразования осуществляются одновременно, т.е. выбирается небольшая часть графа (пересечение ребер, вершина, планарный подграф, имеющий 31 вершину и т.д.), выбирается соответствующая часть булевой функции (которая тоже должна быть локальна), на выбранной части графа осуществляются все этапы преобразования, параллельно все этапы преобразования осуществляются на выбранной части булевой функции, полностью преобразованная часть графа и полностью преобразованная часть булевой функции перемещаются в память, предназначенную только для записи, только после этого в логарифмическую память загружаются новые локальные части графа и функции.

- Логарифмическая память не позволяет хранить последовательность всех пересечений ребер в явном виде и отмечать в ней те пересечения, которые мы уже обработали. Поэтому мы предполагаем, что алгоритм использует естественную дискретизацию плоскости, определяемую начальным списком вершин и ребер графа и, соответственно, не требующую хранения в изменяемой памяти. Алгоритм пользуется естественным алгоритмом сканирования дискретизированной части плоскости, на которой размещается граф. В соответствие с используемой дискретизацией алгоритм в изменяемой памяти хранит только координаты просматриваемого участка дискретизированной части плоскости.
- В рамках первого этапа построения сводимости мы предполагали модификацию вершин дизъюнкций C_i и C_j для каждого пересечения ребер. Хранить эти изменения в явном виде в логарифмической памяти возможности нет. Но в этом нет и необходимости, поскольку то, какие переменные входят в дизъюнкцию, полностью определяется тем, с какими вершинами переменных смежна вершина дизъюнкции.
- В рамках первого этапа построения сводимости предполагается, что пересечения ребер рассматриваются не все по порядку, а

лишь те, у которых отрезки $[\mathcal{W}, x_s]$ и $[\mathcal{W}, x_t]$ не имеют пересечений с ребрами. Заметим, что это, вообще говоря, не создает нагрузку на память, поскольку, как нетрудно убедиться, плоскость можно дискретизировать так, что все пересечения требуемого вида будут просматриваться первыми, а все остальные пересечения будут преобразованы к требуемому виду к моменту их просмотра. В частности, для расположения, используемого в работе [89], подходит прямоугольная дискретизация.

- В ходе преобразований графа появляются новые вершины переменных. Их необходимо включить в цикл вершин переменных. Конструкция, предложенная в работе [89], позволяет изначально запланировать положение всех вершин переменных в цикле, причем позиции в новом цикле как изначально существовавших вершин переменных, так и новых вершин переменных не требуется хранить в памяти, поскольку они могут быть вычислены по начальному списку переменных и дизъюнкций, который расположен в памяти, предназначенной только для чтения (см. [89], рисунок 6).

Перейдем к второму этапу построения сведения. Заметим, что после первого этапа в булевой функции появились дизъюнкции

$$v_1 \vee v_2 \vee v_3 \vee v_4, \tag{30}$$

состоящие из четырех литералов. Следуя [92], заменим каждую подформулу вида (30) на подформулу

$$(v_1 \vee v_2 \vee v_5) \wedge (\neg v_5 \vee v_3 \vee v_4). \tag{31}$$

Легко понять, что функция (30) выполняется тогда и только тогда, когда выполняется функция (31). Однако, в общем случае замена подформулы (30) на подформулу (31) не сохраняет количество решений. В нашем случае каждая подформула вида (30) является подформулой соответствующей формулы (29). Поэтому для каждой формулы вида (30) существует соответствующая формула

$$(\neg v_1 \vee \neg v_2) \wedge (\neg v_2 \vee \neg v_3) \wedge (\neg v_3 \vee \neg v_4) \wedge (\neg v_1 \vee \neg v_4),$$

выполнимость которой возможна лишь в том случае, когда ровно одна из переменных v_1, v_2, v_3, v_4 является истинной. Таким образом, хотя сама по себе замена подформулы (30) на подформулу (31) не обеспечивает сохранения количества решений, эта замена порождает замену подформулы

$$\begin{gathered}(v_1 \vee v_2 \vee v_3 \vee v_4) \wedge \\ (\neg v_1 \vee \neg v_2) \wedge (\neg v_2 \vee \neg v_3) \wedge (\neg v_3 \vee \neg v_4) \wedge (\neg v_1 \vee \neg v_4)\end{gathered} \tag{32}$$

на подформулу

$$\begin{gathered}(v_1 \vee v_2 \vee v_5) \wedge (\neg v_5 \vee v_3 \vee v_4) \wedge \\ (\neg v_1 \vee \neg v_2) \wedge (\neg v_2 \vee \neg v_3) \wedge (\neg v_3 \vee \neg v_4) \wedge (\neg v_1 \vee \neg v_4),\end{gathered} \tag{33}$$

которая гарантирует сохранение количества решений. Замена подформулы (32) на подформулу (33) требует соответствуюшей модификации графа. Необходимо удалить вершину дизъюнкции (30), добавить две вершины новых дизъюнкций $v_1 \vee v_2 \vee v_5$ и $\neg v_5 \vee v_3 \vee v_4$, вершину новой переменной v_5 и соответствующие определению ребра. Полученный граф сохранит планарность [89]. По завершении второго этапа построения сводимости мы получаем некоторую булеву функцию $h(z_1, z_2, \ldots, z_p) = \wedge_{i=1}^{q} H_i$, где для любого i, $1 \leq i \leq q$, функция H_i является дизъюнкцией не более трех литералов. При этом функция h выполняется тогда и только тогда, когда выполняется функция f, замена f на h сохраняет количество решений, граф $G(h)$ является планарным [89, 92].

Перейдем к третьему этапу. Этот этап будет заключительным. Поэтому все данные, прошедшие окончательное преобразование и не требующиеся для дальнейших вычислений, можно размещать в памяти, предназначенной только для записи. Заметим, что для построения сводимости 3SAT $\leq_E$ PC-$\leq$3SAT нам достаточно по функции f построить функцию g с нужными нам свойствами. В частности, функция g должна иметь планарный граф G_g. Однако строить сам граф G_g необходимости нет. Поэтому в память, предназначенную только для записи, мы будем размещать только информацию о функции g. Вся информация о графе G_g будет присутствовать в неявном виде в логарифмической памяти и памяти, предназначенной только для чтения.

Основная цель третьего этапа — удаление из графа всех вершин переменных за счет введения новых вершин дизъюнкций. Мы будем последовательно рассматривать вершины переменных в соответствие с зафиксированным порядком в цикле вершин переменных. Без ограничения общности для некоторого натурального числа $l \in \mathbb{N}$ мы можем обозначать вершины цикла вершин переменных z_i, $1 \leq i \leq l$, предполагая, что сам цикл задается последовательностью $z_1, z_2, \ldots, z_l$. Соответственно, мы можем зафиксировать порядок обхода цикла, предполагая, что вершины посещаются в порядке возрастания номеров.

Выполнение первых двух этапов построения сведения ведет к преобразованию начальной функции f в некоторую функцию h при помощи локальных замен подформул вида (28) на подформулы вида

$$\begin{gathered} C'_i \wedge C'_j \wedge \\ (\neg u_1 \vee \neg u_2 \vee v_1) \wedge (u_1 \vee \neg v_1) \wedge (u_2 \vee \neg v_1) \wedge \\ (\neg u_1 \vee y_t \vee v_2) \wedge (u_1 \vee \neg v_2) \wedge (\neg y_t \vee \neg v_2) \wedge \\ (y_s \vee y_t \vee v_3) \wedge (\neg y_s \vee \neg v_3) \wedge (\neg y_t \vee \neg v_3) \wedge \\ (y_s \vee \neg u_2 \vee v_4) \wedge (\neg y_s \vee \neg v_4) \wedge (u_2 \vee \neg v_4) \wedge \\ (v_1 \vee v_2 \vee v_5) \wedge (\neg v_5 \vee v_3 \vee v_4) \wedge \\ (\neg v_1 \vee \neg v_2) \wedge (\neg v_2 \vee \neg v_3) \wedge (\neg v_3 \vee \neg v_4) \wedge (\neg v_1 \vee \neg v_4) \wedge \\ (u_1 \vee \neg x_s) \wedge (\neg u_1 \vee x_s) \wedge (u_2 \vee \neg x_t) \wedge (\neg u_2 \vee x_t). \end{gathered} \tag{34}$$

На третьем этапе к функции h будут добавляться новые дизъюнкции, уже имеющиеся дизъюнкции будут подвергаться лишь переименованию переменных. Новые имена переменных будут иметь вид $z_{i,j}$, где z_i — вершина переменной, с которой смежна вершина дизъюнкции, а индекс j будет вычисляться по положению ребра на изображении графа. Мы будем предполагать, что в окончательной булевой функции переменные будут упорядочены следующим образом:

$$i_1 < i_2 \Rightarrow z_{i_1,j_1} < z_{i_2,j_2},$$

$$i_1 = i_2, j_1 < j_2 \Rightarrow z_{i_1,j_1} < z_{i_2,j_2}.$$

Кроме того, будем предполагать, что в окончательной булевой функции дизъюнкции из h будут содержать литералы в порядке возрастания переменных. Еще до начала преобразования на третьем этапе мы можем вычислить общее количество дизъюнкций в функции h, количество литералов в каждой дизъюнкции, первые индексы переменных и то, является ли литерал переменной или отрицанием переменной. Всю эту информацию мы можем передать в память, предназначенную только для записи. Таким образом, осуществляя преобразование на третьем этапе, нам нужно будет передать в память, предназначенную только для записи, только вторые индексы переменных функции h и новые дизъюнкции.

Хотя с точки зрения логики последовательного преобразования третий этап является третьим, вычисление на логарифмической памяти предполагает, что он должен вызываться первым и осуществляться как основа всего вычислительного процесса: начинаем обход цикла, если у вершины нет пересекающихся ребер, то преобразуем ее и переходим к следующей вершине, если пересечения есть, то осуществляем одну замену подформулы вида (28) на подформулу вида (34), последовательно преобразуем все новые вершины цикла, передаем информацию в память, предназначенную только для записи, и лишь после этого делаем очередную замену подформулы вида (28) на подформулу вида (34).

Перейдем непосредственно к описанию преобразования третьего этапа. Каждая вершина переменной z_i смежна ровно с двумя другими вершинами других переменных и, вообще говоря с несколькими вершинами дизъюнкций. Без ограничения общности можно предполагать, что ребро (z_i, z_{i+1}) направлено горизонтально влево от вершины z_i, ребро (z_i, z_{i-1}) направлено горизонтально вправо от вершины z_i (для $i = 1$, $i - 1 = l$; для $i = l$, $i + 1 = 1$). Рассмотрим последовательность

$$(z_i, z_{i+1}), e_1, e_2, \ldots, e_p, (z_i, z_{i-1}), e_{p+1}, e_{p+2}, \ldots, e_{p+q}$$

всех ребер, одной из вершин которых является вершина z_i, образованную перечислением ребер на изображении графа по часовой стрелке. Преобразование, соответствующее вершине z_i, заключается в следующей последовательности действий.

- Если вершина дизъюнкции является вершиной ребра e_j, то соответствующая переменная функции h получает обозначение $z_{i,j}$.

Соответственно, после полного завершения третьего этапа все переменные, входящие в h, будут иметь в h единственное вхождение.

- Вычисленные вторые индексы переменных $z_{i,j}$ по мере их вычисления передаются в память, предназначенную только для записи.
- В окончательную функцию добавляется следующая подформула, которая по мере вычисления дизъюнкций передается в память, предназначенную только для записи.

$$\begin{gathered}(u_{i+1} \vee v_{i,1} \vee \neg z_{i,1}) \wedge (z_{i,1} \vee \neg z_{i,2}) \wedge \\ (z_{i,2} \vee \neg z_{i,3}) \wedge \cdots \wedge \\ (z_{i,p-1} \vee \neg z_{i,p}) \wedge (z_{i,p} \vee \neg v_{i,2}) \wedge \\ (v_{i,3} \vee \neg z_{i,p+1}) \wedge (z_{i,p+1} \vee \neg z_{i,p+2}) \wedge \\ (z_{i,p+2} \vee \neg z_{i,p+3}) \wedge \cdots \wedge \\ (z_{i,p+q-1} \vee \neg z_{i,p+q}) \wedge (z_{i,p+q} \vee \neg v_{i,4} \vee u_i) \wedge \neg u_i \wedge \\ (v_{i,4} \vee \neg v_{i,1}) \wedge (v_{i,1} \vee \neg v_{i,2}) \wedge \\ (v_{i,2} \vee \neg v_{i,3}) \wedge (v_{i,3} \vee \neg v_{i,4}),\end{gathered} \tag{35}$$

где $i+1 = 1$ для $i = l$.

Подформула, добавляемая по вершине z_i, $1 \leq i \leq l$, содержит ровно по три вхождения каждой из переменных $v_{i,j}$, $1 \leq j \leq 4$. Переменные $v_{i,j}$, $1 \leq j \leq 4$, содержатся только в формуле (35). Каждая формула вида (35) содержит ровно по два вхождения переменных $z_{i,j}$, $1 \leq j \leq p+q$. Функция h имеет ровно по одному вхождению для каждой такой переменной. Каждая формула вида (35) содержит ровно два вхождения переменной u_i и одно вхождение переменной u_{i+1}. Поэтому получаемая в итоге функция содержит ровно по три вхождения каждой переменной.

Заметим, что в графе G_g ребер между дизъюнкциями из функции h нет, поскольку каждая переменная имеет не более одного вхождения в h. Для подформулы

$$\begin{gathered}(u_{i+1} \vee v_{i,1} \vee \neg z_{i,1}) \wedge (z_{i,1} \vee \neg z_{i,2}) \wedge \\ (z_{i,2} \vee \neg z_{i,3}) \wedge \cdots \wedge \\ (z_{i,p-1} \vee \neg z_{i,p}) \wedge (z_{i,p} \vee \neg v_{i,2})\end{gathered}$$

формулы (35) ребра, соответствующие переменным $z_{i,j}$, можно направить в верхнюю полуплоскость по маршрутам ребер $e_1, e_2, \ldots, e_p$. Два ребра от вершины $u_{i+1} \vee v_{i,1} \vee \neg z_{i,1}$, соответствующие переменной u_{i+1}, можно направить горизонтально влево по маршруту ребра (z_i, z_{i+1}) к дизъюнкциям формулы вида (35), соответствующей вершине z_{i+1}.

Для подформулы

$$\begin{gathered}(v_{i,3} \vee \neg z_{i,p+1}) \wedge (z_{i,p+1} \vee \neg z_{i,p+2}) \wedge \\ (z_{i,p+2} \vee \neg z_{i,p+3}) \wedge \cdots \wedge \\ (z_{i,p+q-1} \vee \neg z_{i,p+q}) \wedge (z_{i,p+q} \vee \neg v_{i,4} \vee u_i) \wedge \neg u_i\end{gathered}$$

формулы (35) ребра, соответствующие переменным $z_{i,j}$, можно направить в нижнюю полуплоскость по маршрутам ребер $e_{p+1}, e_{p+2}, \ldots, e_{p+q}$. Ребра от вершин $z_{i,p+q} \vee \neg v_{i,4} \vee u_i$ и $\neg u_i$, соответствующие переменной u_i, можно направить горизонтально вправо по маршруту ребра (z_i, z_{i-1}) к дизъюнкциям формулы вида (35), соответствующей вершине z_{i-1}. Непосредственной проверкой можно убедиться в том, что дизъюнкции формулы (35), содержащие локальные переменные $v_{i,j}$, можно соединить ребрами без самопересечений. Таким образом, преобразование, связанное с добавлением формул вида (35), позволяет получить планарный граф G_g.

Легко понять, что подграф графа G_g, соответствующий формуле (35), является связным. Переменные u_i обеспечивают принадлежность всех формул вида (35) одной компоненте связности. Поскольку каждая дизъюнкция функции h содержит переменную, которая содержится в некоторой формуле вида (35), граф G_g является связным.

Непосредственной проверкой можно убедиться в том, что выполнимость формулы

$$(v_{i,4} \vee \neg v_{i,1}) \wedge (v_{i,1} \vee \neg v_{i,2}) \wedge (v_{i,2} \vee \neg v_{i,3}) \wedge (v_{i,3} \vee \neg v_{i,4})$$

равносильна выполнимости системы равенств

$$v_{i,1} = v_{i,2} = v_{i,3} = v_{i,4}.$$

Поэтому выполнимость формулы (35) равносильна выполнимости формулы

$$\begin{gathered}(u_{i+1} \vee v_{i,1} \vee \neg z_{i,1}) \wedge (z_{i,1} \vee \neg z_{i,2}) \wedge \\ (z_{i,2} \vee \neg z_{i,3}) \wedge \cdots \wedge \\ (z_{i,p-1} \vee \neg z_{i,p}) \wedge (z_{i,p} \vee \neg v_{i,2}) \wedge \\ (v_{i,3} \vee \neg z_{i,p+1}) \wedge (z_{i,p+1} \vee \neg z_{i,p+2}) \wedge \\ (z_{i,p+2} \vee \neg z_{i,p+3}) \wedge \cdots \wedge \\ (z_{i,p+q-1} \vee \neg z_{i,p+q}) \wedge (z_{i,p+q} \vee \neg v_{i,4} \vee u_i) \wedge \neg u_i \wedge \\ v_{i,1} = v_{i,2} = v_{i,3} = v_{i,4}. \end{gathered} \tag{36}$$

Дизъюнкция $\neg u_i$ может выполняться только при условии $u_i = 0$. Поэтому выполнимость формулы (36) равносильна выполнимости формулы

$$\begin{gathered}(u_{i+1} \vee v_{i,1} \vee \neg z_{i,1}) \wedge (z_{i,1} \vee \neg z_{i,2}) \wedge \\ (z_{i,2} \vee \neg z_{i,3}) \wedge \cdots \wedge \\ (z_{i,p-1} \vee \neg z_{i,p}) \wedge (z_{i,p} \vee \neg v_{i,2}) \wedge \\ (v_{i,3} \vee \neg z_{i,p+1}) \wedge (z_{i,p+1} \vee \neg z_{i,p+2}) \wedge \\ (z_{i,p+2} \vee \neg z_{i,p+3}) \wedge \cdots \wedge \\ (z_{i,p+q-1} \vee \neg z_{i,p+q}) \wedge (z_{i,p+q} \vee \neg v_{i,4}) \wedge \\ v_{i,1} = v_{i,2} = v_{i,3} = v_{i,4}. \end{gathered} \tag{37}$$

Легко проверить, что формула (37) выполняется тогда и только тогда, когда выполняется формула

$$\begin{gathered}(u_{i+1} \vee v \vee \neg z_{i,1}) \wedge (z_{i,1} \vee \neg z_{i,2}) \wedge \\ (z_{i,2} \vee \neg z_{i,3}) \wedge \cdots \wedge \\ (z_{i,p-1} \vee \neg z_{i,p}) \wedge (z_{i,p} \vee \neg v) \wedge \\ (v \vee \neg z_{i,p+1}) \wedge (z_{i,p+1} \vee \neg z_{i,p+2}) \wedge \\ (z_{i,p+2} \vee \neg z_{i,p+3}) \wedge \cdots \wedge \\ (z_{i,p+q-1} \vee \neg z_{i,p+q}) \wedge (z_{i,p+q} \vee \neg v). \end{gathered} \tag{38}$$

Формула вида (35) для вершины z_{i+1} содержит дизъюнкцию $\neg u_{i+1}$, которая может быть выполнима лишь при условии $u_{i+1} = 0$. Поэтому выполнимость формулы (38) равносильна выполнимости формулы

$$\begin{gathered}(v \vee \neg z_{i,1}) \wedge (z_{i,1} \vee \neg z_{i,2}) \wedge \\ (z_{i,2} \vee \neg z_{i,3}) \wedge \cdots \wedge \\ (z_{i,p-1} \vee \neg z_{i,p}) \wedge (z_{i,p} \vee \neg v) \wedge \\ (v \vee \neg z_{i,p+1}) \wedge (z_{i,p+1} \vee \neg z_{i,p+2}) \wedge \\ (z_{i,p+2} \vee \neg z_{i,p+3}) \wedge \cdots \wedge \\ (z_{i,p+q-1} \vee \neg z_{i,p+q}) \wedge (z_{i,p+q} \vee \neg v). \end{gathered} \tag{39}$$

Непосредственной проверкой можно убедиться в том, что формула

$$\begin{gathered}(v \vee \neg z_{i,1}) \wedge (z_{i,1} \vee \neg z_{i,2}) \wedge \\ (z_{i,2} \vee \neg z_{i,3}) \wedge \cdots \wedge \\ (z_{i,p-1} \vee \neg z_{i,p}) \wedge (z_{i,p} \vee \neg v) \end{gathered} \tag{40}$$

выполняется тогда и только тогда, когда выполняется система равенств

$$z_{i,1} = z_{i,2} = \cdots = z_{i,p},$$

а формула

$$\begin{gathered}(v \vee \neg z_{i,p+1}) \wedge (z_{i,p+1} \vee \neg z_{i,p+2}) \wedge \\ (z_{i,p+2} \vee \neg z_{i,p+3}) \wedge \cdots \wedge \\ (z_{i,p+q-1} \vee \neg z_{i,p+q}) \wedge (z_{i,p+q} \vee \neg v) \end{gathered} \tag{41}$$

выполняется тогда и только тогда, когда выполняется система равенств

$$z_{i,p+1} = z_{i,p+2} = \cdots = z_{i,p+q}.$$

Поскольку формула (39) является конъюнкцией формул (40) и (41), отсюда вытекает, что формула (39) выполняется тогда и только тогда, когда выполняется система равенств

$$z_{i,1} = z_{i,2} = \cdots = z_{i,p} = z_{i,p+1} = z_{i,p+2} = \cdots = z_{i,p+q}.$$

Отсюда вытекает, что в функцию h вместо переменных $z_{i,j}$ можно подставить переменную z_i. При этом мы получим равносильность выполнимости функции g и выполнимости функции h. □

Из теоремы 3 получаем следующее утверждение.

Следствие 3. *Проблема* #PC-≤3SAT *является* #**P**-*полной относительно экономной сводимости.*

Хорошо известно, что проблема ≤3SAT остается **NP**-полной даже при ограничении, допускающем использование лишь булевых функций, содержащих ровно три вхождения каждой переменной (см., например, [93], предложение 2.23). Булевы функции, которые мы построили при доказательстве теоремы 3, удовлетворяют этому ограничению. Поэтому в предположении, что каждая булева функция содержит ровно три вхождения каждой переменной, проблема #PC-≤3SAT является #**P**-полной относительно экономной сводимости, а проблема PC-≤3SAT является **NP**-полной относительно экономной сводимости.

Фазовый переход для проблемы выполнимости принято связывать с отношением $\alpha = \frac{m}{n}$, где m — количество дизъюнкций, n — количество переменных [37]. Экспериментальные исследования показывают, что для случайных входов проблемы k-выполнимости существует критическое значение α_c отношения $\frac{m}{n}$ такое, что для $\alpha < \alpha_c$ булева функция с высокой вероятностью выполняется, а для $\alpha > \alpha_c$ функция с высокой вероятностью выполнимой не является. Более того, экспериментальные исследования показывают, что трудные случайные исходные данные имеют близкие к α_c значения α, а по мере удаления от α_c трудность падает [37]. Существование α_c при $n \to +\infty$ для любого k доказано в работе [94]. Для $k = 2$ известно, что $\alpha_c = 1$ [95, 96]. В частности, в работе [97] для $k = 2$ показано, что для фиксированного $\varepsilon > 0$ при $\alpha > 1 + \varepsilon$ и $n \to +\infty$ вероятность невыполнимости булевой функции стремится к 1. Кроме того, для $k = 2$ и фиксированного $\varepsilon > 0$ при $\alpha < 1 - \varepsilon$ и $n \to +\infty$ вероятность выполнимости булевой функции стремится к 1 [97]. В работе [98] установлено, что поведение булевых функций при $k = 2$ аналогично поведению графов. Для $k = 3$ известно, что $\alpha_c \approx 4.26$ [99]. Кроме того, известно, что $3.42 < \alpha_c < 4.506$ [100, 101]. Нетривиальная сложность наблюдается при значениях α, удовлетворяющих соотношениям $3.921 < \alpha_c < 4.256$ [40]. Однако для статистических решателей трудность интервала $3.921 < \alpha_c < 4.256$ начинает проявляться лишь при n существенно больше 1000 [102]. Необходимость учитывать особенности фазового перехода существенно влияет на процесс разработки решателей. Производительность решателей в значительной мере зависит от близости к пороговому значению. В наибольшей степени это влияние сказывается на статистических решателях (см., например, [103, 104, 105]). В последние годы все большее внимание уделяется исследованиям, связанным с структурными характеристиками булевых функций и возможностью их использования для разработки решателей (см., например, [106, 107, 108]). В частности, можно отметить подход, предложенный в работе [109], который основан на поиске логических преобразователей по их представлениям в виде подформул (см. [4, 110, 111]) и

построении графовой структуры. Однако статистические решатели попрежнему играют определяющую роль. Ярким примером является алгоритм WalkSAT (см., например, [112, 113, 114]). Конструкция, рассмотренная при доказательстве теоремы 3, представляется весьма перспективной для эффективной модификации булевых функций в зависимости от требований, связанных с фазовым переходом. Для булевых функций, определяемых сводимостью из доказательства теоремы 3, значение α равно 1, т.е. они являются сильно разреженными. Случайные булевы функции с таким значением α должны выполняться с вероятностью, которая близка к 1. В то же время функции, получаемые при помощи сводимости, содержат всю трудность проблемы SAT. При этом они имеют весьма простую структуру. В частности, трехмерных структур нет вообще. Последовательностью преобразований, например, на основе замен формул вида (20) на формулы вида (21), из этих функций можно получать функции с заданными структурными характеристиками и заданным значением α. В частности, это можно использовать для маскировки и затруднения работы статистических решателей и решателей на основе машинного обучения.

7 Заключение

В данной работе мы установили, что проблема #3SAT является #**P**-полной относительно экономной сводимости в предположении, что в каждой дизъюнкции три литерала соответствуют трем различным переменным. Кроме того, показана #**P**-полнота относительно экономной сводимости для проблем #C-3SAT, #C-≤3SAT, #C-SAT, #SC-3SAT, #PC-≤3SAT. Использование экономной сводимости позволяет получить ряд важных следствий для различных классов вычислительной сложности. В частности, согласно [88], в предположении, что в каждой дизъюнкции три литерала соответствуют трем различным переменным, проблемы $\oplus 3SAT$, $\mathbf{Mod}_k 3SAT$, $\mathbf{Diff}3SAT_{=0}$, $\mathbf{Diff}3SAT_{>0}$, $\mathbf{Diff}3SAT_{=1}$ и $\mathbf{Diff}3SAT_{=g}$ являются полными для классов $\oplus\mathbf{P}$, $\mathbf{Mod}_k\mathbf{P}$, $\mathbf{C}_{=}\mathbf{P}$, **PP**, **SPP** и **WPP**, соответственно.

Трудность проблемы PC-≤3SAT показана для булевых функций с малым значением отношения $\frac{m}{n}$, удобных для модификации и последующего использования в криптографических целях и при создании тестовых данных для решателей.

References

[1] P. Berman, M. Karpinski, A. D. Scott, *Computational complexity of some restricted instances of 3-SAT*, Discrete Applied Mathematics, **155** (2007), 649–653.

[2] A. Darmann, J. Döcker, *On simplified NP-complete variants of Monotone 3-Sat*, Discrete Applied Mathematics, **292** (2021), 45–58.

[3] N. Misra, N. S. Narayanaswamy, V. Raman, B. S. Shankar, *Solving MINONES-2-SAT as Fast as VERTEX COVER*, Lecture Notes in Computer Science, **6281** (2010), 549–555.
[4] G. S. Tseitin, *On the Complexity of Derivation in Propositional Calculus*, Symbolic Computation, **1064** (1983), 466–483.
[5] A. Curtis, J. B. Tenenbaum, T. Lozano-Pérez, L. Kaelbling, *Discovering State and Action Abstractions for Generalized Task and Motion Planning*, Proceedings of the AAAI Conference on Artificial Intelligence, **36** (2022), 5377–5384.
[6] E. Ablad, D. Spensieri, R. Bohlin, J. S. Carlson, A.-B. Strömberg, *Spatial–Temporal Load Balancing and Coordination of Multi-Robot Stations*, IEEE Transactions on Automation Science and Engineering, **20** (2023), 2203–2214.
[7] D. Meli, H. Nakawala, P. Fiorini, *Logic programming for deliberative robotic task planning*, Artifcial Intelligence Review, **56** (2023), 9011–9049.
[8] L. Matlekovic, P. Schneider-Kamp, *Constraint Programming Approach to Coverage-Path Planning for Autonomous Multi-UAV Infrastructure Inspection*, Drones, **7** (2023), 563.
[9] L. Wang, W. Ma, L. Wang, Y. Ren, C. Yu, *Enabling In-Depot Automated Routing and Recharging Scheduling for Automated Electric Bus Transit Systems*, Journal of Advanced Transportation, **2021** (2021), 5531063.
[10] S. Abed, A. Ashkanan, W. Mansoor, A. Gawanmeh, *Enhanced SAT Solvers Based Hashing Method for Bitcoin Mining*, Advances in Intelligent Systems and Computing, **1134** (2020), 191–198.
[11] M. Trimoska, S. Ionica, G. Dequen, *A SAT-Based Approach for Index Calculus on Binary Elliptic Curves*, Progress in Cryptology — AFRICACRYPT 2020, **12174** (2020), 214–235.
[12] S. Utsumi, K. Sakamoto, T. Isobe, *Bit-level evaluation of piccolo block cipher by satisfiability problem solver*, IET Information Security, **17** (2023), 616–625.
[13] C. Huang, M. Newman, M. Szegedy, *Explicit Lower Bounds on Strong Quantum Simulation*, IEEE Transactions on Information Theory, **66** (2020), 5585–5600.
[14] S. Schneider, L. Burgholzer, R. Wille, *A SAT Encoding for Optimal Clifford Circuit Synthesis*, Proceedings of the 28th Asia and South Pacific Design Automation Conference (2023), 190–195.
[15] I. Bliznets, D. Sagunov, K. Simonov, *Fine-grained Complexity of Partial Minimum Satisfiability*, Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, (2022), 1774–1780.
[16] Y. Lin, M. Althoff, *Rule-Compliant Trajectory Repairing using Satisfiability Modulo Theories*, IEEE Intelligent Vehicles Symposium, **4** (2022), 1–8.
[17] D. Caballero, T. Gomez, R. Schweller, T. Wylie, *Unique Assembly Verification in Two-Handed Self-Assembly*, Algorithmica, **85** (2023), 2427–2453.
[18] E. D. Demaine, S. Eisenstat, *Flattening Fixed-Angle Chains Is Strongly NP-Hard*, Lecture Notes in Computer Science, **6844** (2011), 314–325.
[19] F. Imeson, S. L. Smith, *An SMT-Based Approach to Motion Planning for Multiple Robots With Complex Constraints*, IEEE Transactions on Robotics, **35** (2019), 669–684.
[20] J. Giráldez-Cru, J. Levy, *Generating SAT instances with community structure*, Artificial Intelligence, **238** (2016), 119–134.
[21] G. Amendola, F. Ricca, M. Truszczynski, *New models for generating hard random boolean formulas and disjunctive logic programs*, Artificial Intelligence, **279** (2020), 103185.
[22] D. Barak-Pelleg, D. Berend, J. C. Saunders, *A model of random industrial SAT*, Theoretical Computer Science, **910** (2022), 91–112.

[23] R. Heradio, D. Fernandez-Amoros, J. A. Galindo, D. Benavides, D. Batory, *Uniform and scalable sampling of highly configurable systems*, Empirical Software Engineering, **27** (2022), 1–34.
[24] C. Ebert, M. Weyrich, B. Lindemann, S. P. Chandrasekar, *Systematic Testing for Autonomous Driving*, ATZelectronics worldwide, **16** (2021), 18–23.
[25] W. Guo, H.-L. Zhen, X. Li, W. Luo, M. Yuan, Y. Jin, J. Yan, *Machine Learning Methods in Solving the Boolean Satisfiability Problem*, Machine Intelligence Research, **20** (2023), 640–655.
[26] L. Sun, D. Gerault, A. Benamira, T. Peyrin, *NeuroGIFT: Using a Machine Learning Based Sat Solver for Cryptanalysis*, Lecture Notes in Computer Science, **12161** (2020), 62–84.
[27] A. M. Leventi-Peetz, J.-V. Peetz, M. Rohde, *ML Supported Predictions for SAT Solvers Performance*, Advances in Intelligent Systems and Computing, **1069** (2019), 64–78.
[28] M. Mosca, S. R.Verschoor, *Factoring semi-primes with (quantum) SAT-solvers*, Scientifc Reports, **12** (2022), 7982.
[29] P. Branco, P. Mateus, C. Salema, A. Souto, *Using Low-Density Parity-Check codes to improve the McEliece cryptosystem*, Information Sciences, **510** (2020), 243–255.
[30] M. Egger, M. Xhemrishi, A. Wachter-Zeh, R. Bitar, *Sparse and Private Distributed Matrix Multiplication with Straggler Tolerance*, IEEE International Symposium on Information Theory, IEEE, Taipei, 2023.
[31] P. Pérez-Pacheco, P. Caballero-Gil, *McEliece Cryptosystem: Reducing the Key Size with QC-LDPC codes*, 19th International Conference on the Design of Reliable Communication Networks, IEEE, Vilanova i la Geltru, 2023.
[32] S. Afzal, M. Yousaf, H. Afzal, N. Alharbe, M. R. Mufti, *Cryptographic Strength Evaluation of Key Schedule Algorithms*, Security and Communication Networks, **2020** (2020), 3189601.
[33] M. Baldi, J.-C. Deneuville, E. Persichetti, P. Santini, *Cryptanalysis of a Code-Based Signature Scheme Based on the Schnorr-Lyubashevsky Framework*, IEEE Communications Letters, **25** (2021), 2829–2833.
[34] A. K. Hartmann, M. Weigt, *Phase Transitions in Combinatorial Optimization Problems: Basics, Algorithms and Statistical Mechanics*, WILEY-VCH Verlag GmbH & Co. KGaA, Weinheim, 2005.
[35] M. Mézard, A. Montanari, *Information, Physics, and Computation*, Oxford University Press, New York, 2009.
[36] C. Moore, S. Mertens, *The Nature of Computation*, Oxford University Press, New York, 2011.
[37] 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.
[38] 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.
[39] J. Sleegers, R. Olij, G. van Horn, D. van den Berg, *Where the really hard problems aren't*, Operations Research Perspectives, **7** (2000), 100160.
[40] M. Mézard, G. Parisi, R. Zecchina, *Analytic and Algorithmic Solution of Random Satisfiability Problems*, Science, **297** (2002), 812–815.
[41] G. Biroli, S. Cocco, R. Monasson, *Phase transitions and complexity in computer science: an overview of the statistical physics approach to the random satisfiability problem*, Physica A: Statistical Mechanics and its Applications, **306** (2002), 381–394.
[42] W. Perkins, *Searching for (sharp) thresholds in random structures: where are we now?*, arXiv:2401.01800v1 (2024), 1–29.

[43] J. Ding, A. Sly, N. Sun, *Satisfiability Threshold for Random Regular NAE-SAT*, Communications in Mathematical Physics, **341** (2016), 435–489.
[44] J. Ding, A. Sly, N. Sun, *Proof of the satisfiability conjecture for large k*, Annals of Mathematics, **196** (2022), 1–388.
[45] J. Park, H. T. Pham, *A proof of the Kahn–Kalai conjecture*, Journal of the American Mathematical Society, **37** (2024), 235–243.
[46] H. Schawe, R. Bleim, A. K. Hartmann, *Phase transitions of the typical algorithmic complexity of the random satisfiability problem studied with linear programming*, PLoS One, **14** (2019), e0215309.
[47] S. E. Schmittner, *A SAT-based Public Key Cryptography Scheme*, arXiv:1507.08094v3 (2015), 1–11.
[48] R. Sun, Q. Peng, Y. Tian, *A hybrid encryption scheme based on the satisfiability problem*, IEEE Global Conference on Signal and Information Processing, IEEE, Montreal, 2017.
[49] J. Thomas, N. S. Chaudhari, *Hybrid Cryptosystem Based on 2-SAT & 3-SAT and the Implications of Polynomial Solvability of 3-SAT*, International Journal of Computer Applications, **2** (2011), 1–6.
[50] Y. Li, X. Chen, W. Guo, X. Li, W. Luo, J. Huang, H.-L. Zhen, M. Yuan, J. Yan, *HardSATGEN: Understanding the Difficulty of Hard SAT Formula Generation and A Strong Structure-Hardness-Aware Baseline*, Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (2023), 4414–4425.
[51] Z. Newsham, V. Ganesh, S. Fischmeister, G. Audemard, L. Simon, *Impact of Community Structure on SAT Solver Performance*, Lecture Notes in Computer Science, **8561** (2014), 252–268.
[52] A. Slater, *Modelling More Realistic SAT Problems*, Lecture Notes in Computer Science, **2557** (2002), 591–602.
[53] E. D. Demaine, K. Karntikoon, N. Pitimanaaree, *2-Colorable Perfect Matching is NP-complete in 2-Connected 3-Regular Planar Graphs*, arXiv:2309.09786v1 (2023), 1–11.
[54] S. A. Cook, *The complexity of theorem-proving procedures*, Proceedings of the third annual ACM symposium on Theory of computing (1971) 151–158.
[55] M. R. Garey, D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, New York, 1979.
[56] R. M. Karp, *Reducibility among combinatorial problems*, Complexity of Computer Computations, The IBM Research Symposia Series, Springer, Boston, 1972, 85–103.
[57] L. Juban, *Dichotomy theorem for the generalized unique satisfiability problem*, Lecture Notes in Computer Science, **1684** (1999), 327–337.
[58] C. Sundermann, T. Heß, M. Nieke, P. Bittner, J. Young, T. Thüm, I. Schaefer, *Evaluating state-of-the-art # SAT solvers on industrial configuration spaces*, Empirical Software Engineering, **28** (2023), 1–38.
[59] C. Sundermann, T. Thüm, I. Schaefer, *Evaluating #SAT solvers on industrial feature models*, Proceedings of the 14th International Working Conference on Variability Modelling of Software-Intensive Systems, **3** (2020), 1–9.
[60] D. Zhao, L. Liao, W. Luo, J. Xiang, H. Jiang, X. Hu, *Generating Random SAT Instances: Multiple Solutions could be Predefined and Deeply Hidden*, Journal of Artificial Intelligence Research, **76** (2023), 435–470.
[61] R. G. Downey, M. R. Fellows, *Parameterized Complexity*, Springer, New York, 1999.
[62] J. Flum, M. Grohe, *The Parameterized Complexity of Counting Problems*, SIAM Journal on Computing, **33** (2004), 1–29.
[63] R. E. Ladner, N. A. Lynch, A. L. Selman, *A comparison of polynomial time reducibilities*, Theoretical Computer Science, **1** (1975), 103–123.
[64] C. H. Papadimitriou, *Computational Complexity*, Addison-Wesley Publishing Co. Inc., Boston, 1994.

[65] A. Jayasena, P. Mishra, *Directed Test Generation for Hardware Validation: A Survey*, ACM Computing Surveys, **56** (2024), 1–36.
[66] F. Lin, Y. Zhao, *ASSAT: computing answer sets of a logic program by SAT solvers*, Artificial Intelligence, **157** (2004), 115–137.
[67] A. Rubio, R. Cantarero, A. Margara, G. Cugola, D. Villa, J. C. López, *Commonsense reasoning and automatic generation of IoT contextual knowledge: An Answer Set Programming approach*, Internet of Things, **25** (2024), 100998.
[68] T. Soh, N. Tamura, M. Banbara, *Scarab: A Rapid Prototyping Tool for SAT-Based Constraint Programming Systems*, Lecture Notes in Computer Science,, **7962** (2013), 429–436.
[69] D. Kim, N. M. Rahman, S. Mukhopadhyay, *PRESTO: A Processing-in-Memory-Based k -SAT Solver Using Recurrent Stochastic Neural Network With Unsupervised Learning*, IEEE Journal of Solid-State Circuits, (2024), 1–11.
[70] A. A. Sohanghpurwala, M. W. Hassan, P. Athanas, *Hardware accelerated SAT solvers — A survey*, Journal of Parallel and Distributed Computing, **106** (2017), 170–184.
[71] W. Hu, L. Zhang, A. Ardeshiricham, J. Blackstone, B. Hou, Y. Tai, R. Kastner, *Why you should care about don't cares: Exploiting internal don't care conditions for hardware Trojans*, IEEE/ACM International Conference on Computer-Aided Design, IEEE, Irvine, 2017.
[72] D. G. Mahmoud, B. Shokry, V. Lenders, W. Hu, M. Stojilović, *X-Attack 2.0: The Risk of Power Wasters and Satisfiability Don't-Care Hardware Trojans to Shared Cloud FPGAs*, IEEE Access, **12** (2024), 8983–9011.
[73] Y. Su, T. T.-H. Kim, B. Kim, *FlexSpin: A CMOS Ising Machine With 256 Flexible Spin Processing Elements With 8-b Coefficients for Solving Combinatorial Optimization Problems*, IEEE Journal of Solid-State Circuits, (2024), 1–12.
[74] H. Spakowski, M. Thakur, R. Tripathi, *Quantum and classical complexity classes: Separations, collapses, and closure properties*, Information and Computation, **200** (2005), 1–34.
[75] M. Ogiwara, L. A. Hemachandra, *A complexity theory for feasible closure properties*, Journal of Computer and System Sciences, **46** (1993), 295–325.
[76] J. Gill, *Computational complexity of probabilistic Turing machines*, SIAM Journal on Computing, **6** (1977), 675–695.
[77] J. Simon, *On Some Central Problems in Computational Complexity*, Cornell Department of Computer Science Technical Report TR75-224, PhD thesis, Cornell University, Ithaca, 1975.
[78] R. Beigel, J. Gill, *Counting classes: thresholds, parity, mods, and fewness*, Theoretical Computer Science, **103** (1992), 3–23.
[79] J. Cai, L. Hemachandra, *On the power of parity polynomial time*, Mathematical Systems Theory, **23** (1990), 95–106.
[80] U. Hertrampf, *Relations among mod-classes*, Theoretical Computer Science, **74** (1990), 325–328.
[81] L. Goldschlager, I. Parberry, *On the construction of parallel computers from various bases of boolean functions*, Theoretical Computer Science, **43** (1986), 43–58.
[82] C. Papadimitriou, S. Zachos, *Two remarks on the power of counting*, Lecture Notes in Computer Science, **145** (1983), 269–276.
[83] S. Fenner, L. Fortnow, S. Kurtz, *Gap-definable counting classes*, Journal of Computer and System Sciences, **48** (1994), 116–148.
[84] K. W. Wagner, *The complexity of combinatorial problems with succinct input representations*, Acta Informatica, **23** (1986), 325–356.
[85] R. P. N. Rao, J. Rothe, O. Watanabe, *Upward separation for FewP and related classes*, Information Processing Letters, **52** (1994), 175–180.

[86] L. G. Valiant, *A complexity theory for feasible closure properties*, Theoretical Computer Science, **8** (1979), 189–201.
[87] R. Barbanchon, *On unique graph 3-colorability and parsimonious reductions in the plane*, Theoretical Computer Science, **319** (2004), 455–482.
[88] E. Bakali, A. Chalki, S. Kanellopoulos, A. Pagourtzis, S. Zachos, *On the power of counting the total number of computation paths of NPTMs*, arXiv:2306.11614v3 (2024), 1–18.
[89] D. Lichtenstein, *Planar Formulae and Their Uses*, SIAM Journal on Computing, **11** (1982), 329–343.
[90] B. M. E. Moret, *Planar NAE3SAT is in P*, ACM SIGACT News, **19** (1988), 51–54.
[91] M. E. Dyer, A. M. Frieze, *Planar 3DM is NP-complete*, Journal of Algorithms, **7** (1986), 174–184.
[92] H. B. Hunt III, M. V. Marathe, V. Radhakrishnan, R. E. Stearns, *The Complexity of Planar Counting Problems*, SIAM Journal on Computing, **27** (1998), 1–25.
[93] O. Goldreich, *Computational Complexity: A Conceptual Perspective*, Cambridge University Press, Cambridge, 2008.
[94] E. Friedgut, J. Bourgain, *Sharp Thresholds of Graph Properties, and the k-Sat Problem*, Journal of the American Mathematical Society, **12** (1999), 1017–1054.
[95] V. Chvatal, B. Reed, *Mick gets some (the odds are on his side) (satisfiability)*, Proceedings of the 33rd Annual Symposium on Foundations of Computer Science, IEEE, Pittsburgh, 1992.
[96] A. Goerdt, *A threshold for unsatisfiability*, Lecture Notes in Computer Science, **629** (1992), 264–274.
[97] A. Goerdt, *A Threshold for Unsatisfiability*, Journal of Computer and System Sciences, **53** (1996), 469–486.
[98] 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.
[99] J. A. Crawford, L. D. Auton, *Experimental results on the crossover point in random 3-SAT*, Artificial Intelligence, **81** (1996), 31–57.
[100] O. Dubois, Y. Boufkhad, J. Mandler, *Typical random 3-SAT formulae and the satisfiability threshold*, Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms, 2000.
[101] A. C. Kaporis, L. M. Kirousis, E. G. Lalas, *The probabilistic analysis of a greedy satisfiability algorithm*, Random Structures & Algorithms, **28** (2001), 444–480.
[102] M. Mézard, *Optimization and Physics: On the Satisfiability of Random Boolean Formulae*, International Conference on Theoretical Physics, Birkhäuser, Basel, 2003, 475–488.
[103] J. Franco, *Typical case complexity of Satisfiability Algorithms and the threshold phenomenon*, Discrete Applied Mathematics, **153** (2005), 89–123.
[104] H. Fu, S. Cai, G. Wu, J. Liu, X. Yang, Y. Xu, *Improving two-mode algorithm via probabilistic selection for solving satisfiability problem*, Information Sciences, **653** (2024), 119751.
[105] H. Fu, Y. Xu, G. Wu, J. Liu, S. Chen, X. He, *Emphasis on the flipping variable: Towards effective local search for hard random satisfiability*, Information Sciences, **566** (2021), 118–139.
[106] T. N. Alyahya, M. E. B. Menai, H. Mathkour, *On the Structure of the Boolean Satisfiability Problem: A Survey*, ACM Computing Surveys, **55** (2022), 1–34.
[107] 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.
[108] Z. Zhang, D. Xu, J. Zhou, *An algorithm for solving satisfiability problem based on the structural information of formulas*, Frontiers of Computer Science, **15** (2021), 156405.

[109] M. A. H. Newton, M. M. A. Polash, D. N. Pham, J. Thornton, K. Su, A. Sattar, *Evaluating logic gate constraints in local search for structured satisfiability problems*, Artificial Intelligence Review, **54** (2021), 5347–5411.

[110] R. Ostrowski, É. Grégoire, B. Mazure, L Saïs, *Recovering and Exploiting Structural Knowledge from CNF Formulas*, Lecture Notes in Computer Science, **2470** (2002), 185–199.

[111] D. A. Plaisted, S. Greenbaum, *A Structure-preserving Clause Form Translation*, Journal of Symbolic Computation, **2** (1986), 293–304.

[112] S. R. B. Bearden, Y. R. Pei, M. Di Ventra, *Efficient solution of Boolean satisfiability problems with digital memcomputing*, Scientific Reports, **10** (2020), 19741.

[113] S. Cai, C. Luo, K. Su, *Improving WalkSAT By Effective Tie-Breaking and Efficient Implementation*, The Computer Journal, **58** (2015), 2864–2875.

[114] H. Fu, Y. Xu, S. Chen, J. Liu, *Improving WalkSAT for Random 3-SAT Problems*, Journal of Universal Computer Science, **26** (2020), 220–243.

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