Санкт–Петербургский государственный университет

Шарипов Дениль Ильшатович

Выпускная квалификационная работа

Вывод полиномиальных представлений для некоторых вычислительных задач

Бакалавриат:
Направление 01.03.02 «Прикладная математика и информатика»
Основная образовательная программа СВ.5156.2019
«Современное программирование»

Научный руководитель:
доцент, Факультет математики и компьютерных наук СПбГУ, к.ф.-м.н., Шалымов Д. С.

Рецензент:
младший научный сотрудник, Федеральное государственное бюджетное учреждение науки Санкт-Петербургское отделение Математического института им. В.А. Стеклова Российской академии наук, Белова Т. С.

Санкт-Петербург
2024 г.

Содержание

Введение

Доказательство нижних оценок на время работы является центральным направлением теории сложности алгоритмов. К сожалению, это удаётся сделать только в ограниченных моделях вычислений, в общем случае же эта проблема является чрезвычайно сложной. Чтобы как-то совладать со сложностью задачи, была придумана довольно естественная и в то же время мощная идея: доказательство условных нижних оценок. В основу условных нижних оценок входят так называемые тонкие сведения на время работы алгоритмов: сведение $(A, p(n)) \leq (B, q(n))$ означает, что из существования алгоритма для задачи B, который работает „быстрее“ $q(n)$, следует существование более „быстрого“, чем $p(n)$, алгоритма для A. Тривиальным примером тонкого сведения является полиномиальное сведение по Карпу: если NP-трудная задача A сводится к задаче B за полиномиальное время, то в предположении $\mathrm{P} \neq \mathrm{NP}$ задача B не имеет алгоритма с полиномиальным временем работы. Есть довольно большое количество гипотез, которые позволяют доказывать более сильные нижние оценки, например (далее предполагается, что утверждения верны для любого $\varepsilon > 0$):

- ***Задача выполнимости булевых формул*** (***k-SAT***) требует время работы c^n для некоторого неявного $c > 1$ [25], [26];
- ***3-Сумма*** (***3-SUM***) не решается за время $n^{2-\varepsilon}$ [15], [23];
- ***Задача об ортогональных векторах*** (***OV***) не решается за время $n^{2-\varepsilon}$ [40];
- ***Задача о покрытии множествами*** (***Set Cover***) не решается за время $(2-\varepsilon)^n$ [11].
- ***Задача о нахождении попарных кратчайших расстояний***(***APSP***) не решается за время $n^{3-\varepsilon}$ [38].

Одна из самых известных гипотез называется гипотезой сильного экспоненциального времени (SETH). Она утверждает, что для любого $\varepsilon > 0$

существует такое k, что задача ***k-SAT*** не решается за время $(2-\varepsilon)^n$. Тонкие сведения, из которых следуют нижние оценки под гипотезой SETH, называются SETH-сведениями. На данный момент показано довольно много SETH-сведений для различных задач.

Однако построение SETH-сведений для ряда задач является открытой проблемой. В последнем десятилетии были найдены несколько барьеров, которые объясняют сложность построения таких сведений. Первый барьер основан на следующей идее. Пусть $A \in \mathrm{DTIME}[T_1]$, $B \in \mathrm{DTIME}[T_2]$ и A тонко сводится к B таким образом, что если $B \in \mathrm{DTIME}[T_2^{1-\varepsilon}]$ для некоторого $\varepsilon > 0$, то $A \in \mathrm{DTIME}[T_1^{1-\delta}]$ для некоторого $\delta = \delta(\varepsilon)$. Тогда если $B \in (\mathrm{N} \cap \mathrm{coN})\mathrm{TIME}[T_2^{1-\varepsilon}]$, то $A \in (\mathrm{N} \cap \mathrm{coN})\mathrm{TIME}[T_1^{1-\delta}]$ [14, Лемма 3.5]. Таким образом, чтобы исключить сведение $(A, T_1) \leq (B, T_2)$, достаточно показать, что $B \in (\mathrm{N} \cap \mathrm{coN})\mathrm{TIME}[T_2^{1-\varepsilon}]$ для некоторого $\varepsilon > 0$, а A маловероятно принадлежит $(\mathrm{N} \cap \mathrm{coN})\mathrm{TIME}[T_1^{1-\delta}]$ для любого $\delta > 0$. Если в качестве A взять задачу выполнимости булевых формул – ***k-SAT*** – то можно показать барьеры для SETH-сведений. Довольно естественным шагом является введение гипотезы NSETH (сильная гипотеза недетерминированного экспоненциального времени), которая гласит, что ***k-SAT*** $\notin \mathrm{coNTIME}[(2-\varepsilon)^n]$ для любого $\varepsilon > 0$. Оказывается, опровержение NSETH ведет к сильным нижним оценкам на схемную сложность: если NSETH неверна, то класс E^{NP} требует параллельно-последовательные булевы схемы размера $\omega(n)$ [14, Теорема 4.1]. Таким образом, существование SETH-сведений к ряду задач показать не проще, чем доказать сильные нижние оценки на схемную сложность. В предположении NSETH не существует детерминированных SETH-сведений для следующих задач [14, Теорема 5.3] (далее предполагается, что утверждения верны для всех $\varepsilon > 0$; $T(n)$ – время работы, для которого показываем несводимость):

- ***Задача о максимальном потоке*** (***MAXFLOW***) с $T(n) = n^{1+\varepsilon}$;
- ***Задача об ударяющем множестве*** (***HITTING SET***) с $T(n) = n^{1+\varepsilon}$;
- ***3-Сумма*** с $T(n) = n^{1.5+\varepsilon}$;
- ***Задача о нахождении попарных кратчайших расстояний*** с $T(n) =$

$n^{2+\frac{6+\omega}{9}+\varepsilon}$, где ω – наилучшая константа алгоритма произведения матриц.

Совсем недавно был получен ещё один барьер, который показывает сложность доказательства нижних оценок вида λ^n (или λ^k для параметра k) для явных $c > 1$ в предположении SETH [6]. Оказывается, если некоторая NP-трудная задача допускает представление в виде полинома особого вида, то для неё вряд ли существует λ^n-SETH-сведение. Грубо говоря, этот полином должен обладать следующими свойствами:

- $deg(P) = d$ для некоторой константы d;
- P содержит не более $O(2^{n/\theta})$ переменных для некоторой константы θ;
- Каждая переменная может быть вычислена за время $O(2^{n/\theta'})$ для некоторой константы θ';
- X yes-экземпляр задачи тогда и только тогда, когда $P(\phi(X)) > 0$, где ϕ – отображение подстановки значений в переменные.

Статья [6] приводит такие представления для различных NP-трудных как непараметризованных, так и параметризованных задач, и тем самым показывается сложность доказательства нижних оценок для этих задач вида c^n или c^k под SETH: ***k-SAT***, ***MAX-k-SAT***, ***Гамильтонов путь***, ***Раскраска графа***, ***Задача о покрытии множествами***, ***Независимое множество***, ***Клика***, ***Вершинное покрытие***, ***3d-Паросочетание***, ***k-Путь***, ***k-Вершинное покрытие***, ***k-Дерево***, ***k-Изменение кластера*** и другие.

В этой работе мы обобщаем результат [6] на $k^{O(1)}$-SETH-сведения (эта запись означает, что сведение показывает нижнюю оценку $k^{O(1)}$ на время работы, где k – некоторый параметр) и выводим полиномиальные представления для более широкого круга задач (как NP-трудных, так и для полиномиальных). Основной результат работы представлен следующими теоремами.

Теорема 1. *Пусть существует хотя бы одно из следующих сведений ($\lambda > 1$):*

- *λ^n-SETH-сведение к задаче **Наименьшая общая надстрока**;*
- *λ^n-SETH-сведение к задаче **Наименьшее общее разбиенее строк** (n – длина строки);*

- *$\lambda^{n\frac{\log\log n}{\log n}}$-SETH-сведение к задаче* ***Наименьшее общее разбиение строк с константным алфавитом*** *(λ^k-SETH-сведение к задаче с параметром $k = n\frac{\log\log n}{\log n}$);*

- *λ^k-SETH-сведение к задаче* ***k-Мотив*** *графа, где k – размер мотива;*

Тогда выполняется как минимум одно из следующих утверждений:

- *Класс E^{NP} не вычисляется семейством булевых параллельно-последовательных схем размера $O(n)$;*

- *Для любой $\gamma > 1$ существует явный полином константной степени, который не вычисляется арифметическими схемами размера n^γ.*

Теорема 2. *Пусть существует хотя бы одно из следующих сведений ($q > 1$):*

- *n^q-SETH-сведение к задаче* ***k-Сумма с константным k****;*

- *$(n+m)^q$-SETH-сведение к задаче* ***H-Изоморфизм подграфа с константным H*** *(k^q-SETH-сведение к задаче с параметром $k = n + m$);*

Тогда выполняется как минимум одно из следующих утверждений:

- *Класс E^{NP} не вычисляется семейством булевых параллельно-последовательных схем размера $O(n)$;*

- *Для некоторой $\gamma > 1$ существует явный полином константной степени, который не вычисляется арифметическими схемами размера n^γ.*

Постановка задачи

Задача данной работы заключается в следующих пунктах:

1. Изучить концепцию полиномиальных представлений и их применение для установления барьеров для SETH-сведений ([6]).

2. Изучить существующие полиномиальные представления для различных NP-трудных задач из статьи [6] и используемые для их вывода идеи и инструменты.

3. Обобщить результат теоремы 4.4 [6] на $k^{O(1)}$-SETH-сведения, где k – некоторый параметр задачи.

4. Использовать полученные результаты для построения достаточно эффективных полиномиальных представлений как для более широкого круга NP-трудных задач, так и для полиномиальных задач.

5. Установить барьеры для SETH-трудности этих задач. Доказать Теоремы 1 и 2.

1. Обозначения

Для двух множеств A и B обозначим $A \sqcup B$ их дизъюнктное объединение. Для некоторого положительного числа n обозначим $[n] = \{1, \ldots, n\}$. Для графа $G(V, E)$, вершины $v \in V$ и множества вершин $S \subseteq V$ обозначим $N(G, v)$ множество соседей вершины v, $N(G, S) = \cup_v N(G, v)$. Под $O^*(.)$ будем скрывать полиномиальный фактор, т.е. $O^*(T(n)) = O(T(n)n^{O(1)})$. В контексте сложности полиномиальных представлений, а также в контексте схемной сложности для простоты записи в некоторых местах будем опускать полиномиальные факторы для $O(c^k n^{O(1)})$ и логарифмические факторы для $O(k^{O(1)} \log^{O(1)} n)$. Для задач с параметром k будем считать, что $k \geq \omega(\log |x|)$, где x – экземпляр задачи.

2. Обзорный раздел по предметной области

В этом разделе мы дадим все необходимые определения и условия теорем.

2.1. Тонкие сведения на время работы алгоритмов

Термин тонкого сведения впервые появляется в работах Райана и Вирджинии Вильямс. Например, в работе [38] вводится понятие *субкубического сведения* – прообраза тонкого сведения. Каноническое определение тонкого сведения приводится в [36]. Тем не менее, мы будем использовать немного дополненное определение из [6].

Определение 1 (Тонкие сведения на время работы алгоритмов, [6])**.** Пусть A и B это некоторые вычислительные задачи, $a, b\colon \mathbb{Z}_{\geq 0} \to \mathbb{Z}_{\geq 0}$ некоторые неубывающие функции и $\delta\colon \mathbb{R}_{>0} \to \mathbb{R}_{>0}$ Скажем, что $(A, a(n))$ *δ-тонко сводится* к $(B, b(n))$ (в формальной записи $(A, a(n)) \leq_\delta (B, b(n))$), если для любого $\varepsilon > 0$ существует алгоритм Q для A с оракульным доступом к B, константа c, функция $t(n)\colon \mathbb{Z}_{\geq 0} \to \mathbb{Z}_{\geq 0}$, такая что на любом экземпляре A размера n алгоритм Q

- имеет время работы не более $ca(n)^{1-\delta(\varepsilon)}$;
- создает не более $t(n)$ экземпляров B адаптивно: каждый экземпляр зависит от предыдущих созданных экзмепляров, как и ответы оракула на них;
- размеры n_i созданных экземпляров удовлетворяют неравенству

$$\sum_{i=1}^{t(n)} b(n_i)^{1-\varepsilon} \leq ca(n)^{1-\delta}\,.$$

Нас интересует частный случай тонких сведений – SETH-сведения, которые мы определим далее.

2.2. Задача выполнимости и гипотеза SETH

Определение 2 (Гипотеза сильного экспоненциального времени (SETH), [26, 25])**.** Для любого $\varepsilon > 0$ существует k, такое что

$$\textbf{\textit{k-SAT}} \notin \mathrm{DTIME}[2^{(1-\varepsilon)n}].$$

Определение 3 (Недетерминированная SETH (NSETH), [14], [25])**.**
Для любого $\varepsilon > 0$ существует k, такое что

$$\textbf{\textit{k-TAUT}} \notin \mathrm{NTIME}[2^{(1-\varepsilon)n}].$$

Определение 4 (SETH-трудность для экспоненциальных задач, [6])**.**
Для некоторой константы $\lambda > 1$ скажем, что задача A с параметром k λ^k-*SETH-трудная*, если существует функция $\delta\colon \mathbb{R}_{>0} \to \mathbb{R}_{>0}$, и для всех $q \in \mathbb{N}$ и $\varepsilon > 0$ существует алгоритм Q для задачи ***q-SAT*** с оракульным доступом к A, функция $t(n)\colon \mathbb{Z}_{\geq 0} \to \mathbb{Z}_{\geq 0}$ и константа $d \in \mathbb{N}$, такие что на любом экземпляре задачи ***q-SAT*** размера n алгоритм A

- имеет время работы не более $O(2^{(1-\delta(\varepsilon))n})$;
- создает не более $t(n)$ экземпляров B адаптивно: каждый экземпляр зависит от предыдущих созданных экзмепляров, как и ответы оракула на них;
- размеры n_i и параметры k_i созданных экземпляров удовлетворяют неравенству

$$\sum_{i=1}^{t(n)} \lambda^{(1-\varepsilon)k_i} \cdot n_i^d \leq O(2^{(1-\delta)n})\,.$$

Если задача A с параметром k λ^k-SETH-трудная, то любой алгоритм для A с временем работы $O(\lambda^k n^d)$ решает ***k-SAT*** за время $O(2^{(1-\delta(\varepsilon))n})$ для всех k, тем самым опровергая гипотезу SETH.

Определение 5 (SETH-трудность для полиномиальных задач)**.**
Для некоторой константы $q > 1$ скажем, что задача A с параметром k k^q-*SETH-трудная*, если существует функция $\delta\colon \mathbb{R}_{>0} \to \mathbb{R}_{>0}$, и для всех $\varepsilon >$

0 существует алгоритм Q для задачи ***q-SAT*** с оракульным доступом к A, функция $t(n)\colon \mathbb{Z}_{\geq 0} \to \mathbb{Z}_{\geq 0}$ и константа $d \in \mathbb{N}$, такие что на любом экземпляре задачи ***q-SAT*** размера n алгоритм A

- имеет время работы не более $O(2^{(1-\delta(\varepsilon))n})$;
- создает не более $t(n)$ экземпляров B адаптивно: каждый экземпляр зависит от предыдущих созданных экзмепляров, как и ответы оракула на них;
- размеры n_i и параметры k_i созданных экземпляров удовлетворяют неравенству

$$\sum_{i=1}^{t(n)} k_i^q \log^d n_i \leq O(2^{(1-\delta)n}) \,.$$

Если задача A с параметром k k^q-SETH-трудная, то любой алгоритм для A с временем работы $O(k^q \log^d n)$ решает ***k-SAT*** за время $O(2^{(1-\delta(\varepsilon))n})$ для всех k, тем самым опровергая гипотезу SETH.

2.3. Нижние оценки на схемную сложность

Несмотря на то, что понятия булевых и арифметических схем появились еще во второй половине 20-го века, нам будет удобно пользоваться определениями из [6].

Определение 6 (Булева схема, [6])**.** Булева схема C с переменными $x_1, \ldots, x_n$ – это ориентированный ацикличный граф в котором

- каждая вершина (которая называется гейтом) имеет входящую степень 0 или 2;
- вершины с входящей степенью 0 помечены либо переменными x_i, либо константами 0 или 1;
- вершины с входящей степенью 2 помечены булевыми функциями $\{0,1\} \to \{0,1\}$;

- единственная вершина с исходящей степенью 0 называется выходом схемы.

Заметим, что гейт, вычисляющий унарную функцию отрицания, может быть представлен в бинарном как $g \oplus 1$, где g – гейт-родитель. Булева схема C с переменными $x_1, \ldots, x_n$ вычисляет некоторую булеву функцию $f\colon \{0,1\} \to \{0,1\}$. Размер схемы мы определим как количество гейтов в ней, сложность булевой функции определим как минимальный размер схемы, которая ее вычисляет.

Определение 7 (Параллельно-последовательная булева схема, [6])**.** Схема называется *параллельно-последовательной*, если существует такая нумерация ℓ гейтов схемы, что для любого провода (u, v) выполняется $\ell(u) < \ell(v)$, и ни для какой пары проводов $(u, v), (u', v')$ не выполняется $\ell(u) < \ell(u') < \ell(v) < \ell(v')$.

Наилучшая известная нижняя оценка на на размер булевых схем для функций из P составляет $3.1n - o(n)$ [29]. Та же оценка является наилучшей известной и в ограниченной модели параллельно-последовательных булевых схем для гораздо более широкого класса функций E^{NP}. Открытой задачей уже много лет является нахождение явной функции (то есть функции из P или NP), которая бы не вычислялась булевыми схемами линейного размера из различных ограниченных классов схем [35, 2].

Открытая задача 1 ([14, 6])**.** Найти функцию в классе E^{NP}, которая не вычисляется параллельно-последовательными схемами линейного размера.

Определение 8 (Арифметическая схема, [6])**.** Арифметическая схема C с переменными $x_1, \ldots, x_n$ над кольцом R – это неориентированный ациклический граф, в котором

- каждая вершина (которая называется гейтом) имеет входящую степень 0 или 2;

- вершины с входящей степенью 0 помечены либо переменными x_i, либо элементами R;

- вершины с входящей степенью 2 помечены либо операцией $+$, либо операцией $\times$;

- каждый гейт с исходящей степенью 0 называется выходным гейтом.

Далее мы будем использовать $R = \mathbb{Z}$ или $R = \mathbb{Z}_p$ для некоторого простого p. Скажем, что арифметическая схема C вычисляет полином $P(x_1, \ldots, x_n)$, если полином, который вычисляет схема C, совпадает с P (то есть коэффициенты при каждом мономе в этих полиномах равны). Определим размер C как количесто проводов (ребер) в ней, арифметическую схемную сложность полинома как минимальный размер схемы, которая вычисляет его.

Известно, что полином $x_1^r + \ldots + x_n^r$ над полем F требует для вычисления арифметические схемы размера $\Omega(n \log r)$ (если r не делит характеристику поля F) ([34, 10]).

Определение 9 (Явное семейство полиномов)**.** Назовем семейство полиномов $\mathcal{P} = \{P_n\}_{n \in \mathbb{N}}$ *явным* (Δ-*явным*), если существует константа Δ, такая что для всех n степень P_n не превосходит Δ, абсолютное значение всех коэффициентов не превосходит $O(n^\Delta)$ и все они могут быть вычислены одновременно за время $O(n^\Delta)$.

Открытая задача 2. Предъявить явный полином (семейство полиномов), который требует для вычисления арифметические схемы размера $n^{1+\varepsilon}$ для некоторого $\varepsilon > 0$.

2.4. Полиномиальные представления

Определение 10 (Полиномиальные представления для экспоненциальных задач, [6])**.** Пусть A – параметризованная задача с параметром k, Δ – константа, $T\colon \mathbb{N} \to \mathbb{N}$ – неубывающая функция, такая что $T(k) = 2^{\omega(\log k)}$, $\mathcal{P} = \{P_n\}_{n \in \mathbb{N}}$ – Δ-явное семейство полиномов над $\mathbb{Z}$. Скажем, что $\mathcal{P}$ – Δ-явное полиномиальное представление A сложности T, если существуют

- функция $s\colon A \times \mathbb{N}$, такая что $s(x,k) \leq T(k)|x|^\Delta$, и алгоритм, вычисляющий $s(x,k)$ за время $T(k)|x|^\Delta$;

- функция $\phi\colon A \times \mathbb{N} \to \mathbb{Z}^*$, такая что $\phi(x,k) \in \mathbb{Z}^{s(x,k)}$, и алгоритм, вычисляющий $\phi(x,k)$ за время $T(k)|x|^{\Delta}$,

такие что выполняется следующее. Для любых $(x,k) \in A \times \mathbb{N}$

- $P_{s(x,k)}(\phi(x,k)) \neq 0 \Leftrightarrow (x,k)$ это yes-экземпляр задачи A ;
- $|P_{s(x,k)}(\phi(x,k))| < 2^{s(x,k)^{o(1)}}$.

Определение 11 (Полиномиальные представления для полиномиальных задач)**.** Пусть A – параметризованная задача с параметром k, Δ – константа, $T\colon \mathbb{N} \to \mathbb{N}$ – неубывающая функция, такая что $T(k) = k^{O(1)}$, $\mathcal{P} = \{P_n\}_{n\in\mathbb{N}}$ – Δ-явное семейство полиномов над $\mathbb{Z}$. Скажем, что $\mathcal{P}$ – Δ-явное полиномиальное представление A сложности T, если существуют

- функция $s\colon A \times \mathbb{N}$, такая что $T(k)\log^{\Delta}|x|$, и алгоритм, вычисляющий $s(x,k)$ за время $T(k)\log^{\Delta}|x|$;
- функция $\phi\colon A \times \mathbb{N} \to \mathbb{Z}^*$, такая что $\phi(x,k) \in \mathbb{Z}^{s(x,k)}$, и алгоритм, вычисляющий $\phi(x,k)$ за время $T(k)\log^{\Delta}|x|$,

такие что выполняется следующее. Для любых $(x,k) \in A \times \mathbb{N}$

- $P_{s(x,k)}(\phi(x,k)) \neq 0 \Leftrightarrow (x,k)$ это yes-экземпляр задачи A ;
- $|P_{s(x,k)}(\phi(x,k))| < 2^{s(x,k)^{o(1)}}$.

В качестве функции s далее мы будем использовать канторовскую нумерацию:

$$s(x,k) = (|x|+k)(|x|+k+1)/2 + k.$$

2.5. Задача нахождения арифметической схемы минимального размера

Для доказательства Теоремы 4 нам понадобится недетерминированно находить арифметическую схему минимального размера, вычисляющую некоторый полином.

Определение 12 (***Gap-MACP***, [6])**.** Пусть p – простое число, $\mathcal{P} = \{P_n\}_{n\in\mathbb{N}}$ – явное семейство полиномов, c – некоторая константа. Задача $\mathrm{Gap\text{-}MACP}_{\mathcal{P},c,p}(n,s)$ заключается в следующем: по данным $n, s \in \mathbb{N}$ найти арифметическую схему над $\mathbb{Z}_p$ размера не более cs, вычисляющую P_n над $\mathbb{Z}_p$, либо если такой схемы не существует вывести что угодно.

Лемма 1 (Лемма 3.5 из [6])**.** *Существует константа $\mu > 0$, такая что для любого Δ-явного семейства полиномов $\mathcal{P}$ и любого простого числа p*

$$\mathrm{Gap\text{-}MACP}_{\mathcal{P},\mu\Delta^2,p}(n,s) \in \mathrm{NTIME}[O(sn^{2\Delta}\log^2 p)]\,.$$

Доказательство. Пусть полином P степени Δ вычисляется арифметической схемой размера s. Тогда P вычисляется *гомогенной* арифметической схемой размера $O(\Delta^2 s) = O(s)$: все гейты этой схемы вычисляют полиномы степени не более Δ. Недетерминированно угадаем такую схему. Ее можно проверить довольно легко, последовательно вычисляя полиномы в каждом гейте за время $n^{2\Delta}\log^2 p$ и сравнив результат на выходе с P. □

2.6. Разбиение дерева на сбалансированные поддеревья

Для построения полиномиального представления для задачи ***k-Мотив графа*** нам понадобится следующая лемма и определение.

Лемма 2 (Лемма 6.1, [6])**.** *Пусть $T(V,E)$ – дерево, $M \subseteq V$ – множество из k вершин и $\theta > 1$ – некоторое число. Существует разбиение E на $m \le \theta$ блоков $E = E_1 \sqcup \ldots \sqcup E_m$, такое что для каждого $i \in [m]$ E_i индуцирует поддерево T_i дерева T с не более чем $2k/(\theta-1)+2$ вершинами из M.*

Определение 13 (Определение 6.2, [6])**.** Для данных m множеств $X_1, \ldots, X_m$ введем множество $S = \{s_1, \ldots, s_m\}$, такое что S не пересекает никакое из X_i, и множество $C = \bigcup\limits_{i\neq j}(X_i \cap X_j)$. *Графом подмножеств* $B(X_1, \ldots, X_m)$ назовем следующий граф $G(V,E)$: $V = S \sqcup C$, $E = \{\{s_i, c\} \colon c \in X_i \cap C\}$. Каждая вершина $v \in S$ называется *вершиной-множеством* и вершина $v \in C$ называется *вершиной-коннектором*.

Пусть $T_1, \ldots, T_m$ – деревья, получившиеся после применения Леммы 2 к дереву T. Тогда $B = B(V(T_1), \ldots, V(T_m))$ представляет собой дерево, состоящее из m вершин-множеств и не более $m - 1$ вершин-коннекторов.

2.7. Задачи, которые рассматриваются в работе

Задача ***Наименьшая общая надстрока*** находит широкое применение в биоинформатике [39, 32] и сжатии данных [21, 17, 33]. ***Наименьшая общая надстрока*** NP-трудна [21] и MAX-SNP-трудна [7], но допускает константное приближение за полиномиальное время. Задача известна своей знаменитой жадной гипотезой, которая утверждает, что простой жадный алгоритм для нее дает 2-приближение [7]. ***Наименьшая общая надстрока*** довольно просто решается за время $O^*(2^n)$ с помощью сведения к задаче нахождения гамильтонова пути максимального веса в ориентированном графе. Однако до сих пор неизвестно, можно ли решить ее за время $O^*(c^n)$ для некоторого $c < 2$ [20, 19]. В статье [18] поднимается вопрос о возможной SETH-трудности ***Наименьшей общей надстроки***.

Задача ***Наименьшее общее разбиение строк*** NP-трудна и APX-трудна [22]. Наилучшая верхняя оценка на сложность ***Наименьшего общего разбиения строк*** составляет $O^*(1.618^n)$[12]. Неизвестно, точная ли это оценка или нет, как и вопрос о том, SETH-трудная ли эта задача. Задача ***Наименьшее общее разбиение строк с константным алфавитом*** в свою очередь может быть решена за субэкспоненциальное время $2^{O(n \log\log n/\log n)}$ [12]. Известно, что она не решается за время $2^{o(n/\log n)}$ в предположении гипотезы ETH. Вопрос о том, является ли нижняя оценка $2^{O(n/\log n)}$ точной или ее можно усилить является открытым. Также остается открытым вопрос, можно ли уменьшить время $2^{O(n \log\log n/\log n)}$.

Задача ***k-Мотив графа*** также находит применение в биоинформатике [28, 8]. Задача является NP-трудной, а также FPT, если взять параметр равный размеру мотива [16]. Наилучшая верхняя оценка на детерминированную параметризованную сложность составляет $O^*(5.22^k)$ [31]. Известно, что задача SCH-трудна: алгоритм, работающий за время $(2 - \varepsilon)^n$ опроверг бы гипотезу SCH (гипотеза о том, что ***Задача о покрытии множествами*** не решается

за время $(2-\varepsilon)^n$). В работе [9] изучается сложность ***k-Мотива графа*** относительно различных структурных параметров. Однако не известно, является ли задача 2^k-SETH-трудной относительно размера мотива.

Задача ***k-Сумма с константным k*** широко изучалась с точки зрения тонких оценок на время работы алгоритмов. Например, знаменитая гипотеза 3-SUM гласит, что нельзя решить задачу поиска трех чисел в массиве с суммой 0 за время $n^{2-\varepsilon}$ [24]. В работе [14] доказывается, что задача ***3-Сумма*** не является n^2-SETH-трудной в предположении гипотезы SETH. При этом остается открытым вопрос, является ли она $n^{1.5}$-SETH-трудной, а также являются ли n^q-SETH-трудными для какого-нибудь $q > 1$ задачи ***4-Сумма*** или ***5-Сумма*** или в общем случае ***k-Сумма с константным k***.

Задачу ***H-изоморфизм подграфа с константным H*** можно переформулировать, как задачу поиска некоторого фиксированного шаблона H в графе. Частным случаем этой задачи является задача поиска ***k-Клики с константным k*** в графе. Гипотеза k-Clique гласит, что эту задачу нельзя решить за время $n^{\frac{\omega k}{3}-\varepsilon}$, где ω - наилучшая константа перемножения матриц [37]. Неизвестно, является ли эта оценка точной под SETH.

3. Основной результат

3.1. Полиномиальные представления для NP-трудных задач

Теорема 3. *[6] Пусть A – некоторая вычислительная задача с параметром k. Предположим, что для любой $c > 1$ существует $\Delta = \Delta(c) > 0$, такая что A допускает Δ-полиномиальное представление сложности c^k. Тогда если A λ^k-SETH-трудная для некоторой константы $\lambda > 1$, то выполняется как минимум одно из следующих утверждений:*

- *Класс E^{NP} не вычисляется семейством булевых параллельно-последовательных схем размера $O(n)$;*
- *Для любой $\gamma > 1$ существует явное семейство полиномов константной степени, которое не вычисляется арифметическими схемами размера n^γ.*

Для доказательства Теоремы 1 необходимо показать существование Δ-полиномиальных представлений сложности c^k для любых $c > 1$.

Лемма 3. *Для любой $c > 1$:*

- *задача **Наименьшая общая надстрока** допускает $\Delta = \Delta(c)$-полиномиальное представление сложности c^n (n – количество строк);*
- *задача **Наименьшее общее разбиение строк** допускает $\Delta = \Delta(c)$-полиномиальное представление сложности c^n (n – длина строки);*
- *задача **Наименьшее общее разбиение строк с константным алфавитом** допускает $\Delta = \Delta(c)$-полиномиальное представление сложности $c^{n\frac{\log\log n}{\log n}}$;*
- *задача **k-Мотив графа** допускает $\Delta = \Delta(c)$-полиномиальное представление сложности c^k (k – размер мотива).*

Доказательство. Для удобства опишем полиномиальные представления в следующем формате:

Идея. Краткое описание идеи, которая стоит за полиномиальным представлением.

Переменные. Описание переменных, которые участвуют в представлении.

Сложность. Подсчет сложности полиномиального представления: за какое время можно вычислить значения для всех переменных?

Полином. Непосредственно сам полином или его описание.

Степень. Подсчет степени представления. Удостоверяемся, что степень является константой.

Наименьшая общая надстрока. Дано n строк $s_1, \ldots, s_n$ с длинами $|s_i| \leq n^{O(1)}$. Определить, существует ли строка длины не больше k, содержащая в качестве подстрок $s_1, \ldots, s_n$.

Идея. Давайте считать, что среди строк $s_1, \ldots, s_n$ нет такой пары s_i, s_j, что s_i подстрока s_j или s_j подстрока s_i (если, например, s_i является подстрокой s_j, то можно удалить s_i из нашего набора, от этого ответ не изменится).

Нетрудно заметить, что как минимум одна из подходящих надстрок получается путём последовательной конкатенации строк $s_{i_1}, \ldots, s_{i_r}$ для некоторых индексов $i_1, \ldots, i_r$ с возможными перекрытиями соседних пар (например, набор строк aa, ab, bc имеет общую надстроку длины 3, которая равна $abc = a \cdot overlap(aa, ab) \cdot overlap(ab, bc) \cdot c$, где $\cdot$ обозначает конкатенацию строк, а $overlap(u, v)$ – наибольший суффикс u, совпадающий с префиксом v). Заметим также, что если мы добавляем к частично построенной надстроке некоторую строку s_j и последняя добавленная строка это s_i, то длина перекрытия s_j с надстрокой строго меньше $|s_i|$ (иначе s_i была бы подстрокой s_j). Разобьем последовательность строк $s_{i_1}, \ldots, s_{i_r}$ на θ блоков B_i размера $\leq \lceil n/\theta \rceil$. Пусть $first_i$ – первая строка из блока B_i, $last_i$ – последняя, S_i – общая надстрока, полученная последовательной конкатенацией с перекрытиями всех

строк из блока B_i. Исходную надстроку можно получить последовательной конкатенацией с перекрытиями надстрок S_i. Заметим, что при фиксированных $first_i$ и $last_i$, S_i – наименьшая общая надстрока строк из блока B_i. Пусть $ss(M_i, first_i, last_i)$ – наименьшая общая надстрока строк с индексами из M_i, префикс которой совпадает с $first_i$ и суффикс совпадает с $last_i$. Тогда в полиноме можно перебрать все $first_i$ и $last_i$, множества M_i индексов строк при фиксированных $first_i$ и $last_i$ и проверить, что $\sum |ss(M_i, first_i, last_i)| - \sum |overlap(last_i, first_{i+1})| = q$.

Переменные. Введем $s(n) = O\left(\binom{n}{\leq\lceil n/\theta\rceil})n^{O(1)}\right) = O^*\left(\binom{n}{\lceil n/\theta\rceil}\right)$ переменных:

$$X = \{x_{M,f,l,q} \colon S \subseteq [n], 1 \leq |M| \leq n/\theta, f, l \in S, 0 \leq q \in\leq n^{O(1)}\}.$$

$$Y = \{y_{f,l,q} \colon f, l \in [n], 0 \leq q \leq n^{O(1)}\}.$$

$$Z = \{z_q \colon 0 \leq q \leq n\}.$$

Отображение $\phi_{s(n)}(I)$ устанавливает следующие значения переменным:

$x_{M,f,l,q} = 1$, если длина наименьшей общей надстроки строк с индексами из M, в префиксе которой находится строка s_f и в суффиксе которой находится строка s_l, равна q; $x_{M,f,l,q} = 0$, иначе.

$y_{f,l,q} = 1$, если $|overlap(s_f, s_l)| = q$; $y_{f,l,q} = 0$, иначе.

$z_q = 1$, если $q = k$; $z_q = 0$, иначе.

Сложность. Сложность представления составляет $O^*\left(\binom{n}{\lceil n/\theta\rceil}2^{\lceil n/\theta\rceil}\right)$, поскольку количество переменных $y_{f,l,q}$ составляет $n^{O(1)}$ и их значения вычисляются тоже за $n^{O(1)}$, количество переменных $x_{S,f,l,q}$ составляет $O^*\left(\binom{n}{\lceil n/\theta\rceil}\right)$ и их значения вычисляются за время $O^*\left(2^{\lceil n/\theta\rceil}\right)$ следующим образом. Сведем задачу к задаче нахождения гамильтонова пути максимального веса построением следующего ориентированного графа G: вершинами будут индексы строк (V = M), каждая пара вершин соединена ориентированным ребром и ребро (u, v) для некоторых вершин u и v имеет вес равный $|overlap(s_u, s_v)|$. Гамильтонов путь максимального веса можно найти методом динамического программирования: Пусть

для $S \subseteq V, v \in V$ $dp[S][v]$ хранит максимальный вес гамильтонова пути в графе $G[S]$, заканчивающегося в вершине v и начинающегося в вершине f. Опишем базу динамического программирования. Изначально присвоим $dp[S][f] = 0$ для $S = \{f\}$ (где f – индекс строки, которая должа быть в префиксе наименьшей общей надстроки строк с индексами из M). Опишем переход.

$$dp[S][v] = \max_{u \in V, u \neq v} (dp[S \setminus \{v\}][u] + w(u, v)).$$

Максимальным весом гамильтонова пути, начинающегося в вершине f и заканчивающегося в вершине l будет $dp[V][l]$. Путь, который соответствует этому весу, является искомой наименьшей общей настрокой строк с индексами из M. Время работы пересчёта составляет $O\left(n^2 2^{\lceil n/\theta \rceil}\right) = O^*\left(2^{n/\theta}\right)$.

Полином. Полином $P_{s(n)}$ определяется следующим образом.

Для всех наборов $S_1, \ldots, S_\theta \subseteq [n]$, таких что $1 \leq |S_i| \leq \lceil n/\theta \rceil$, $S_i \cap S_j = \varnothing$ при $i \neq j$ и $\cup_i S_i = [n]$, для всех наборов $f_1, \ldots, f_\theta, l_1, \ldots, l_\theta \in [n]$, таких что $f_i, l_i \in S_i$, $f_i \neq l_i$ если $|S_i| > 1$, для всех $0 \leq q \leq n$, наборов $0 \leq q_1, \ldots, q_\theta, r_1, \ldots, r_{\theta-1} \leq n^{O(1)}$, таких что $q_1 + \ldots + q_\theta - r_1 - \ldots - r_{\theta-1} = q$, добавим моном

$$z_q \prod_{i \in [\theta]} x_{S_i, f_i, l_i, q_i} \prod_{i \in [\theta-1]} y_{l_i, f_{i+1}, r_i}.$$

Степень. Степень полинома составляет $2\theta + 1$.

Наименьшее общее разбиение подстрок. Даны две n-символьных строки a и b над алфавитом Σ и число $k \leq n$. Определить, можно ли разрезать строку a на k непрерывных подстрок таким образом, чтобы их перестановкой можно было получить строку b.

Идея. Рассмотрим искомое разбиение a. Подстроки из разбиения можно распределить по θ множествам $A_1, \ldots, A_\theta$ размера $\leq \lceil n/\theta \rceil$ (в данном случае под подстрокой мы подразумеваем элемент $[n] \times [n]$ (начало и конец),

а не саму строку). Поскольку каждая подстрока соответствует некоторой подстроке b, A_i соответствует некоторому множеству подстрок b B_i размера $|A_i|$. Тогда в полиноме можно перебрать все возможные наборы множеств подстрок $A_1, \ldots, A_\theta$ и $B_1, \ldots, B_\theta$ размера $\leq \lceil n/\theta \rceil$ и проверить, что подстроки A_i совпадают с соответствующими подстроками B_i для каждого i, а также, что $\sum |A_i| = k$.

Переменные. Введем $s(n) = O\left(\binom{2n}{\leq 2\lceil n/\theta \rceil}^2\right) = O^*\left(\binom{2n}{2\lceil n/\theta \rceil}^2\right)$ переменных (концы $\lceil n/\theta \rceil$ отрезков можно выбрать не более чем $\binom{2n}{2\lceil n/\theta \rceil}$ способами):

$$X = \{x_{A,B} \colon A \subseteq [n] \times [n], B \subseteq [n] \times [n], |A| = |B| \leq \lceil n/\theta \rceil,$$

$$\forall (i,j) \in A, B \ i \leq j, \forall (i_1, j_1), (i_2, j_2) \in A, B \ j_1 < i_2 \ or \ j_2 < i_1\}.$$

$$Y = \{y_t \colon 0 \leq t \leq n\}.$$

Отображение $\phi_{s(n)}(I)$ устанавливает следующие значения переменным:

$x_{A,B} = 1$, если мультимножество подстрок a, заданных отрезками из A, совпадает с мультимножеством подстрок b, заданных отрезками из B; $x_{A,B} = 0$, иначе.

$y_t = 1$, если $t = k$; $y_t = 0$, иначе.

Сложность. Сложность составляет $O^*\left(\binom{2n}{2\lceil n/\theta \rceil}^2\right)$, поскольку мультимножества длинных кусков можно сравнить за полиномиальное время.

Полином. Полином $P_{s(n)}$ определяется следующим образом. Для всех $t \leq n$, для всех пар множеств $A, B \subseteq [n] \times [n]$ попарно непересекающихся отрезков, таких что $|A| = |B| = t$ и сумма длин отрезков в A и B равна n, для всех дизъюнктных разбиений $A_1 \cup \ldots \cup A_\theta = A$ и $B_1 \cup \ldots \cup B_\theta = B$, таких что $|A_i| = |B_i| \leq \lceil n/\theta \rceil$ добавим моном

$$\prod_{i \in [\theta]} z_t x_{A_i, B_i}.$$

Степень. Степень $\mathcal{P}$ составляет 2θ.

Заметим, что похожую формулировку можно построить для любого константного числа строк.

Наименьшее общее разбиение подстрок с константным алфавитом. Даны две n-символьных строки a и b над алфавитом Σ константного размера и число $k \leq n$. Определить, можно ли разрезать строку a на k непрерывных подстрок таким образом, чтобы их перестановкой можно было получить строку b.

Идея. Введём некоторый параметр $\mu = o(n)$, значение которого мы определим позже. В любом корректном разбиении a (то есть в разбиении, подстроки которого можно переставить и получить b) есть длинные строки (размера $> \mu$) и короткие строки (размера $\leq \mu$). Длинных подстрок не более $\lfloor n/\mu \rfloor$, поэтому их можно разбить на θ мультимножеств размера не более $\lceil n/\mu\theta \rceil$. Тогда можно переберать все возможные такие наборы мультимножеств $A_1^1, \ldots, A_\theta^1$ для строки a, такие что $|A_i^1| \leq \lceil n/\mu\theta \rceil$ и $B_1^1, \ldots, B_\theta^1$ для строки b, такие что $|A_i^1| = |B_i^1| \leq \lceil n/\mu\theta \rceil$. Далее необходимо проверить для каждого i, совпадают ли A_i и B_i, что делается за полиномиальное время.

Теперь остается разбить остатки строк a и b на короткие подстроки. Для этого заметим, что если вырезать $\lfloor n/\mu \rfloor$ подстрок из строки, то количество оставшихся подстрок не будет превышать $\lfloor n/\mu \rfloor + 1 \leq 2\lfloor n/\mu \rfloor$. Тогда можно снова перебрать разбиения этих подстрок по θ мультимножествам $A_1^2, \ldots, A_\theta^2$ и $B_1^2, \ldots, B_\theta^2$ размера не более $2\lceil n/\mu\theta \rceil$. Воспользуемся определением гистограммы для мультимножества строк S [Теорема 8, Minimum Common String Partition: Exact Algorithms]: $hist_\mu(S) = \{(s, count(S, s)) \| \ s \in \Sigma^*, |s| \leq \mu\}$, где $count(S, s)$ - число вхождений s в S. Пусть $hist_\mu(n)$ – множество всех возможных гистаграмм, в которых каждая строка имеет длину не более μ и имеет не более n вхождений. Переберем все возможные гистограммы $H \in hist_\mu(n)$, а также разбиения $H_1^A \cup \ldots \cup H_\theta^A = H$ и $H_1^B \cup \ldots \cup H_\theta^B = H$ (объединение гистограмм Q и R определяется следующим образом: для любой пары $(s_q, k_q) \in Q$, $(s_r, k_r) \in R$, такой что $s_q = s_r$, $(s_q, k_q + k_r) \in Q \cup R$; если же s_q не встре-

чается в R, то $(s_q, k_q) \in Q \cup R$, аналогично с s_r). Теперь для каждого множества A_i^2 и B_i^2 необходимо проверить, что H_i^A является возможной гистограммой разбиения A_i^2 и H_i^B является возможной гистограммой разбиения B_i^2. В процессе перебора необходимо отсекать конфигурации мультимножеств длинных кусков и наборов гистограмм, суммарное число строк в которых превосходит k.

Переменные. Введем $s(n) = O\left(\binom{2n}{\leq 2\lceil n/\mu\theta\rceil}^2 + 2n^{|\Sigma|^\mu}\binom{2n}{\leq 4\lceil n/\mu\theta\rceil}\right) = O^*\left(\binom{2n}{4\lceil n/\mu\theta\rceil}^2\right)$ переменных (концы $\lceil n/\mu\theta\rceil$ отрезков можно выбрать не более чем $\binom{2n}{2\lceil n/\mu\theta\rceil}$ способами и количество различных гистаграмм не превосходит $n^{|\Sigma|^\mu}$):

$$X = \{x_{A,B} \colon A \subseteq [n] \times [n], B \subseteq [n] \times [n], |A| = |B| \leq \lceil n/\mu\theta\rceil,$$

$$\forall (i,j) \in A, B\ i \leq j, \forall (i_1,j_1), (i_2,j_2) \in A, B\ j_1 < i_2\ or\ j_2 < i_1\}.$$

$$Y = \{y_{l,S,H} \colon l \in \{1,2\}, S \subseteq [n] \times [n], |S| \leq 2\lceil n/\mu\theta\rceil,$$

$$\forall (i,j) \in S\ i \leq j, \forall (i_1,j_1), (i_2,j_2) \in S\ j_1 < i_2\ or\ j_2 < i_1, H \in hist_\mu(n)\}.$$

$$Z = \{z_t \colon 0 \leq t \leq n\}.$$

Отображение $\phi_{s(n)}(I)$ устанавливает следующие значения переменным:

$x_{A,B} = 1$, если мультимножество подстрок a, заданных отрезками из A, совпадает с мультимножеством подстрок b, заданных отрезками из B; $x_{A,B} = 0$, иначе.

$y_{l,S,H} = 1$, если H – одна из возможных гистограмм мультимножества подстрок a, если $l = 0$, или подстрок b, если $l = 1$, заданных отрезками из S; $y_{S,H,l} = 0$, иначе.

$z_t = 1$, если $t = k$; $z_t = 0$, иначе.

Сложность. Сложность составляет $O^*\left(\binom{2n}{4\lceil n/\mu\theta\rceil}^2 n^{|\Sigma|^\mu}\right)$, поскольку мультимножества длинных кусков можно сравнить за полиномиальное время, число различных гистограмм не превосходит $n^{|\Sigma|^\mu}$ и найти все гистограммы мультимножества строк можно за то же время, используя метод динамического программирования [Теорема 8, [12]]. Подстав-

ляя $\mu = \alpha \log n / \log |\Sigma|$, увеличивая θ, мы можем добиться времени $O^*\left(2^{\frac{\log\log n}{\theta' \log n}}\right)$ для любой константы θ'.

Полином. Для множества попарно непересекающихся отрезков $S \subseteq [n] \times [n]$ обозначим за $\hat{S} \subseteq [n] \times [n]$ такое множество попарно непересекающихся отрезков, что для любой пары $o_1 \in S$ и $o_2 \in \hat{S}$ o_1 не пересекается с o_2 и суммарная длина отрезков в $S \cup \hat{S}$ равна n (по сути, $\hat{S}$ является дополнением S).

Полином $P_{s(n)}$ определяется следующим образом. Для всех t, r, таких что $t + r \leq n$, для всех пар множеств $S_a, S_b \subseteq [n] \times [n]$ попарно непересекающихся отрезков длины $\geq \mu$, таких что $|S_a|, |S_b| = r$, для всех дизъюнктных разбиений $A_1 \cup \ldots \cup A_\theta = S_a$ и $B_1 \cup \ldots \cup B_\theta = S_b$, таких что $|A_i| = |B_i| \leq \lceil n/\mu\theta \rceil$, для всех дизъюнктных разбиений $S_1 \cup \ldots \cup S_\theta = \hat{S}_a$, такого что $|S_i| \leq \lceil 2n/\mu\theta \rceil$, для любой гистограммы $H \in hist_\mu(n)$ с суммарным числом вхождений не более t, для всех $H_1^A \cup \ldots \cup H_\theta^A = H$ и $H_1^B \cup \ldots \cup H_\theta^B = H$, добавим моном

$$\prod_{i \in [\theta]} z_t x_{A_i, B_i} y_{1, S_i, H_i^A} y_{2, S_i, H_i^B}.$$

Степень. Степень $\mathcal{P}$ составляет 4θ.

Заметим, что похожую формулировку можно построить для любого константного числа строк.

k-Мотив графа. Дан граф $G(V, E)$ на n вершинах, раскраска вершин $c : V \to \mathcal{C}$ и мультимножество M цветов из $\mathcal{C}$. Необхожимо определить, существует ли подмножество вершин $R \subseteq V$, такое что граф $G[R]$ связен и $c(R) = M$, где $c(R)$ – мультимножество цветов вершин из R.

Идея. Идея полиномиального представления для задачи ***k-Мотив графа*** очень похожа на идею полиномиального представления для задачи ***k-Путь*** из статьи [6]. В этой задаче необходимо найти простой путь в графе на n вершинах, состоящий из k вершин. Классическим методом решения ***k-Пути*** является техника цветного кодирования, которая

была представлена в работе [5]: раскрасим каждую вершину графа в случайный цвет $c \in [k]$; тогда с вероятностью e^{-k} все вершины пути длины k будут раскрашены в разные цвета; по фиксированной раскраске вершин можно найти такой цветной путь за время $2^k n^{O(1)}$ с помощью метода динамического программирования. Поскольку мы не можем использовать случайные биты, необходимо дерандомизировать данный алгоритм. Это можно сделать с помощью k-совершенного семейства F хеш-функций $f\colon [n] \to [k]$.

Семейство хеш-функций $\mathcal{F}$ называется k-совершенным, если для любого множества $S \subseteq [n]$ размера $|S| = k$ найдется функция $f \in \mathcal{F}$, которая инъективна на S (в терминах цветного кодирования раскрашивает вершины S в разные цвета). Работа [6] предъявляет такое семейство $\mathcal{F}$ размера $O^*(e^{2k/\theta}) = O^*(3^{2k/\theta})$, состоящее из функций $f\colon [n] \to [\theta k]$ и вычисляемое за время $O^*(2^{9k})$. Чтобы решить задачу ***k-Путь*** детерминированно за время $2^{O(k)}$, достаточно перебрать все такие функции из $\mathcal{F}$, и для каждой такой функции-раскраски искать цветной путь с помощью метода динамического программирования, как и в вероятностном случае.

Полиномиальное представление же для ***k-Пути*** использует довольно простое наблюдение: путь из k вершин, раскрашенный функцией $f\colon [n] \to [\theta k]$, можно разбить на θ кусков размера не более $\lceil k/\theta \rceil$, каждый из которых использует не более $\lceil k/\theta \rceil$ различных цветов. Таким образом, достаточно верифицировать лишь наличие каждого куска пути по отдельности, а также верифицировать их связность. Для этого вполне хватит $O^*(c^{k/\theta})$ переменных для некоторой константы $c > 1$.

Полиномиальное представление для ***k-Мотива графа*** устроено аналогичным образом. В любом связном подграфе графа G размера k можно найти остовное дерево. Это дерево можно разбить на θ поддеревьев T_i размера не более $\lceil 3k/\theta \rceil$ с помощью Леммы 2. Для этих поддеревьев можно перебрать часть мотива $c(T_i)$, которую он занимает в $c(T)$; обязательное условие, что $c(T) = c(T_1) \sqcup \ldots \sqcup c(T_\theta)$. Теперь в полиноме необходимо верифицировать наличие каждого поддерева T_i с мотивом

$c(T_i)$, а также связность поддеревьев между собой. Это можно сделать с помощью того же семейства k-совершенных хеш-функций $\mathcal{F}$.

Переменные. Введём

$$s(n) = O\left(3^{\lceil 2k/\theta \rceil}\binom{\theta k}{\leq \lceil 3k/\theta \rceil}\binom{k}{\leq \lceil 3k/\theta \rceil}n^{O(1)}\right) = O^*(3^{\lceil 2k/\theta \rceil}\binom{\theta k}{\lceil 3k/\theta \rceil}^2)$$

переменных:

$$X = \{x_{f,T,C,H} \colon f \in \mathcal{F}, T \subseteq [\theta k], |T| \leq \lceil 3k/\theta \rceil,$$

$$C \subseteq [k], |C| \leq \lceil 3k/\theta \rceil, H \subseteq V, |H| \leq \theta\}.$$

$$Y = \{y_{f,v,t,r} \colon f \in \mathcal{F}, v \in V, t \in [\theta k], r \in [k]\}.$$

Отображение $\phi_{s(n)}(I)$ устанавливает следующие значения переменным:

$x_{f,T,C,H} = 1$, если в G существует связное множество вершин $W \subseteq V \setminus H$, такое что $f(W) = T$, $c(W) = M[C]$ и $H \subseteq N(G, W)$; $x_{f,T,C,H} = 0$, иначе.

$y_{f,v,t,r} = 1$, если $f(v) = t$ и $c(v) = M[r]$; $y_{f,v,t,r} = 0$, иначе.

Сложность. Покажем, как вычислить значение каждой переменной из X за время $O^*\left(2^{6k/\theta}\right)$ с помощью метода динамического программирования. Пусть $v \in V$, $T \subseteq [\theta k]$, $C \subseteq [k]$, $H \subseteq [\theta]$. Пусть $dp[u][T'][C'][H'] = 1$, если в G существует связное множество вершин $W \subseteq V \setminus H$, такое что $u \in W$, $f(W) = T'$, $c(W) = M[C']$, $H \subseteq N(G, W)$ и $dp[v][T][C][H] = 0$, иначе. Опишем базу динамического программирования. Изначально присвоим $dp[u][f(u)][\{j\}][A] = 1$ для всех вершин u, всех $j \in [k]$, таких что $M[j] = c(u)$ и всех множеств $A \subseteq (N(G, u) \cap H)$. Всем остальным состояниям dp присвоим 0. Опишем переход. Пусть необходимо пересчитать состояние $dp[u][T'][C'][H']$. Если $f(u) \notin T'$ или $c(u) \notin C'$, то $dp[u][T'][C'][H'] = 0$. Иначе, $dp[u][T'][C'][H'] =$

$$\bigvee_{i \in N(G,u), j \in [k] : M[j] = c(u)} dp[i][T' \setminus f(u)][C' \setminus \{j\}][H' \setminus (N(G, u) \cap H')].$$

Когда все состояния dp посчитаны, можно вычислить значение переменной: $x_{f,T,C,H} = \bigvee\limits_{u \in V \setminus S} dp[v][T][C][S]$. Время работы пересчёта составляет $O\left(n 2^{2\lceil 3k/\theta \rceil}\right) = O^*\left(2^{6k/\theta}\right)$.

Сложность представления составляет

$$O^*\left(2^{6k/\theta} 3^{\lceil 2k/\theta \rceil} \binom{\theta k}{\lceil 3k/\theta \rceil}^2)\right) = O^*\left(2^{11k/\theta} \binom{\theta k}{\lceil 3k/\theta \rceil}^2\right),$$

поскольку значение каждой переменной из X вычисляется за время $O^*\left(2^{6k/\theta}\right)$, а значение каждой переменной из Y вычисляется за время $O^*(1)$.

Полином. Полином $P_{s(n),k}$ можно построить следующим образом. Переберём все функции $f \in \mathcal{F}$ Переберём все $m \leq \theta$ и все наборы $T_1, \ldots, T_m \subseteq [\theta k]$, такие что $|T_i \cap T_j| \leq 1$ для любых $i \neq j$ и граф $B(T_1, \ldots, T_m)$ связен и является деревом, $|T_i| \leq \lceil 3k\theta \rceil$ и $|\cup_i T_i| = k$. Пусть Q – множество вершин-коннекторов графа $B = B(T_1, \ldots, T_m)$ (это в точности множество попарных пересечений T_i), а $S = \{s_1, \ldots, s_m\}$ – множество вершин-множеств ($|Q|, |S| \leq m$). Переберём все наборы $C_1, \ldots, C_\theta \subseteq [k]$ (множества индексов мотивов), такие что $|\cup_i C_i| = k$, $|C_i \cap C_j| \leq 1$ и $|T_i \cap T_j| = 1 \Leftrightarrow |C_i \cap C_j| = 1$ для любых $i \neq j$. Пересечения C_i сопоставляют индекс мотива каждому коннектору из Q. Пусть $r(q)$ для $q \in Q$ обозначает соответствующий q индекс мотива. Переберём все возможные назначения $v(q)$ вершин из V для $q \in Q$, такие что для $q_1 \neq q_2$ $v(q_1) \neq v(q_2)$. Добавим моном

$$\prod_{q \in Q} y_{f,v(q),q,r(q)} \prod_{i \in [m]} x_{f,\hat{T}_i,\hat{C}_i,H_i},$$

где $\hat{C}_i = C_i \setminus \{r(q) : q \in (Q \cap N(B, s_i))\}$, $\hat{T}_i = T_i \setminus (Q \cap N(B, s_i))$, $H_i = \{v(q) : q \in (Q \cap N(B, s_i))\}$.

Степень Степень полинома не превосходит 2θ.

□

Теорема 1. *Пусть существует хотя бы одно из следующих сведений ($\lambda > 1$):*

- *λ^n-SETH-сведение к задаче **Наименьшая общая надстрока**;*
- *λ^n-SETH-сведение к задаче **Наименьшее общее разбиенее строк** (n – длина строки);*
- *$\lambda^{n\frac{\log\log n}{\log n}}$-SETH-сведение к задаче **Наименьшее общее разбиение строк с константным алфавитом** (λ^k-SETH-сведение к задаче с параметром $k = n\frac{\log\log n}{\log n}$);*
- *λ^k-SETH-сведение к задаче **k-Мотив** графа, где k – размер мотива;*

Тогда выполняется как минимум одно из следующих утверждений:

- *Класс E^{NP} не вычисляется семейством булевых параллельно-последовательных схем размера $O(n)$;*
- *Для любой $\gamma > 1$ существует явный полином константной степени, который не вычисляется арифметическими схемами размера n^γ.*

Доказательство. Теорема непосредственно следует из Теоремы 3 и Леммы 3. □

3.2. Полиномиальные представления для полиномиальных задач

Теорема 4. *Пусть A – некоторая вычислительная задача с параметром k, r и $q > 1$ – некоторые константы, такие что $r < q$. Предположим, что существует $\Delta = \Delta(r) > 0$, такая что A допускает Δ-полиномиальное представление сложности k^r. Тогда если A k^q-SETH-трудная, то выполняется как минимум одно из следующих утверждений:*

- *Класс E^{NP} не вычисляется семейством булевых параллельно-последовательных схем размера $O(n)$;*
- *Для некоторой $\gamma > 1$ существует явное семейство полиномов константной степени, которое не вычисляется арифметическими схемами размера n^γ.*

Доказательство. Зафиксируем параметры $\gamma > 1$ и $\alpha < 1$, которые определим позже.

Пусть $\mathcal{P} = (P_t)_{t \geq 1}$ – это семейство Δ-полиномиальных представлений для задачи A сложности k^q.

Предположим, что A – k^q-SETH-трудная задача, значит существует такая функция $\varepsilon \colon \mathbb{R}_{>0} \to \mathbb{R}_{>0}$, что для любой $t \in N$ $(\textbf{\textit{t-SAT}}, 2^n) \leq_\varepsilon (A, k^q)$. Покажем, что выполняется хотя бы одна из двух нижних оценок на схемы.

Если $\mathcal{P}$ не вычисляется семейством арифметических схем над $\mathbb{Z}$ размера n^γ, $\gamma > 1$, тогда мы получаем явное семейство полиномов константной степени, которое не вычисляется арифметическими схемами размера n^γ (параметр γ будет определен позже). Пусть теперь $\mathcal{P}$ вычисляется семейством арифметических схем над $\mathbb{Z}$ размера n^γ. Мы покажем, что гипотеза NSETH в таком случае неверна (т.е. для ***t-TAUT*** существует алгоритм с временем работы $2^{(1-\varepsilon)n}$ для любого t), из чего следует суперлинейная нижняя оценка на размер булевых схем, вычисляющих языки из класса E^{NP}.

Заметим, что для того чтобы проверить t-DNF формулу на тавтологичность, достаточно проверить ее отрицание - t-CNF формулу - на невыполнимость. Таким образом мы сводим задачу ***t-TAUT*** к ***t-UNSAT***, преобразуя t-DNF формулы в t-CNF. Используя свойство самосводимости SAT, сведем t-CNF формулу к $2^{(1-\alpha)n}$ формулам на αn переменных с помощью расщепления по $(1-\alpha)n$ переменным.

Используя тонкое SETH-сведение, сведем получившиеся экземпляры ***t-UNSAT*** к экземплярам задачи A с параметром $k \leq L$, где $L \leq 2^{\alpha n/q}$. Далее сосредоточимся на решении этих экземпляров.

По условию для задачи A существет семейство Δ-полиномиальных представлений $\mathcal{P}$ сложности k^r, то есть существуют $s \colon A \times \mathbb{N} \to \mathbb{N}$, $s(x,k) \leq k^r \log^{O(1)} n \leq L^r \log^{O(1)} n \leq 2^{\alpha \frac{r}{q} n + o(n)} =\leq 2^{(\alpha \frac{r}{q} + o(1))n}$ (для достаточно большого n) и $\phi \colon A \times \mathbb{N} \to \mathbb{Z}^*$ (вычислимые за время $2^{2\alpha \frac{r}{q} n}$), что для любого $x \in A$ и $k \in \mathbb{N}$,

- $P_{s(x,k)}(\phi(x,k)) \neq 0 \Leftrightarrow (x,k)$ это yes экземпляр A ;
- $|P_{s(x,k)}(\phi(x,k))| < 2^{s(x,k)^{o(1)}} < 2^{(k \log n)^{o(1)}}$.

Пусть $T = 2^{(\alpha\frac{r}{q}+o(1))n}$. Найдем простое число $p \leq max|P_{s(x,k)}(\phi(x,k))| < 2^{(k\log n)^{o(1)}}$ недетерминированно за время $O(p^7)$ [1, 27]. Поскольку семейство $\mathcal{P}$ явное, каждый полином P_t можно выписать за время $O(t^\Delta \log^2 p) = O(T^{\Delta+1})$ для $t \leq T$. Теперь достаточно вычислить значения P_t в точках, равных экземплярам задачи A. Для этого угадаем маленькие арифметические схемы, вычисляющие полиномы P_t для всех $t \leq T$. Будем рассматривать каждый полином P_t в кольце $\mathbb{Z}_{p_t}$ (назовем его Q_t), что дает нам гарантию того, что при вычислении значения полинома числа не вырастут сильно. Обозначим $\mathcal{Q} = (Q_1, Q_2, \ldots)$. Далее решим задачу $\text{Gap-MACP}_{\mathcal{Q},\mu\Delta^2,p}(t, ct^\gamma)$ для всех $t \leq T$, используя Лемму 1. Поскольку мы предположили, что для $\mathcal{P}$ существует семейство арифметических схем размера ct^γ, для $\mathcal{Q}$ тоже будет существовать семейство схем размера ct^γ, его вычисляющее. Отсюда следует, что для $\mathcal{Q}$ есть семейство гомогенных арифметических схем C_t размера не более $\mu\Delta^2 t^\gamma$. По лемме 1 задача $\text{Gap-MACP}_{\mathcal{Q},\mu\Delta^2,p}(t, ct^\gamma)$ решается недетерминированно за время

$$O\left(\Delta^2 \cdot ct^\gamma \cdot t^{2\Delta} \cdot \log^2(p)\right) = O\left(T^{\gamma+2\Delta+1}\right).$$

Поскольку необходимо угадать арифметическую схему C_t для каждого $t \leq T$, суммарное время работы шага получения схем составляет

$$O\left(p^7 + T(T^{\Delta+1} + T^{\gamma+2\Delta+1})\right) = O\left(T^{\gamma+2\Delta+10}\right) = O\left(T^{\gamma+2\Delta+10}\right).$$

Теперь остается подставить все $2^{(1-\alpha)n}$ экземпляров задачи A в схемы C_t и вычислить результат. Этот можно выполнить за время

$$O(2^{(1-\alpha)n} T^\gamma \log^2 p) = O(2^{(1-\alpha)n} T^{\gamma+o(1)}).$$

Итоговое время работы алгоритма вместе с шагом 3 составляет

$$O(T^{\gamma+2\Delta+10} + 2^{(1-\alpha)n} T^{\gamma+o(1)}) =$$

$$O(2^{(\alpha\frac{r}{q}+o(1))n(\gamma+2\Delta+10)} + 2^{(1-\alpha+(\alpha\frac{r}{q}+o(1))(\gamma+o(1)))n}).$$

Подберем α и γ таким образом, чтобы оба слагаемых были меньше

$2^{(1-\varepsilon)n}$ для некоторого $\varepsilon > 0$. Возьмем $\alpha = \frac{1}{10(r/q)(\gamma+2\Delta+10)}$. Тогда при достаточно большом n первое слагаемое превратится в $2^{n/10}$. Возьмем $\gamma = 1+\frac{1}{2}(\frac{q}{r}-1) = \frac{1}{2}+\frac{1}{2}\frac{q}{r}$. Поскольку $q > r$, $\frac{q}{r} - 1 > 0$ и $\gamma > 1$. Тогда при достаточно большом n второе слагаемое превращается в $2^{(1-\frac{1}{2}-\frac{1}{2}\frac{q}{r}+(\frac{1}{2}+\frac{1}{2}\frac{q}{r})\frac{r}{q})n} = 2^{(1-\frac{1}{2}(\frac{q}{r}-\frac{r}{q}))n}$. Поскольку $\frac{q}{r} > 1 > \frac{r}{q}$, слагаемое равно $2^{(1-\delta)n}$ для $\delta = \frac{1}{2}(\frac{q}{r}-\frac{r}{q}) > 0$. Получаем недетерминированный алгоритм с временем работы $2^{(1-\varepsilon)n}$ для $\varepsilon > 0$, тем самым опровергая гипотезу NSETH и получая нижние оценки на схемную сложность. □

Для доказательства Теоремы 2 остается лишь показать существование Δ-полиномиальных представлений сложности k^r для нужных r.

Лемма 4. *Для любого* $1 < r < 2$

- *задача* ***k-Сумма с константным k*** *допускает полиномиальное представление константной степени* $\Delta = \Delta(r)$ *и сложности* n^r*;*
- *задача* ***H-Изоморфизм подграфа с константным H*** *допускает полиномиальное представление константной степени* $\Delta = \Delta(r)$ *и сложности* $(n+m)^r$.

Доказательство. Опишем представления в том же формате, что и в Лемме 3.

k-Сумма с константным k. Дано k массивов A_i длины n (k – константа), необходимо выбрать по элементу в каждом (числа в промежутке $[-n^c, n^c]$, где c – константа), чтобы их сумма равнялась 0.

Идея. Сначала рассмотрим наивное представление. Введем переменные $X_{i,j,t}$ для $i \in [k]$, $j \in [n]$, $t \in [-n^c, n^c]$, которые принимают значение 1 в том случае, если $A_i[j] = t$. Тогда достаточно проверить существование номеров элементов $j_1, \ldots, j_k$ и чисел $t_1, \ldots, t_k$, таких что $t_1 + \ldots + t_k = 0$ и $\prod_{i=1}^{k} X_{i,j_i,t_i} = 1$. Проблема в том, что число переменных составляет $k \cdot n \cdot 2n^c = O(n^{c+1})$, что может быть сильно больше $n^{\lceil k/2 \rceil}$. Можно обойти эту проблему следующим образом. Заметим, что битовая длина чисел не превосходит $L = \lceil 2c \log n \rceil$ (будем считать, что j-тый бит числа равен нулю, если $j > L$). Пусть $\pi_{\theta,l}(x)$ обозначает число x, у которого

оставили биты с номерами (начиная с нуля) от $l\lceil\frac{L}{\theta}\rceil$ до $(l+1)\lceil\frac{L}{\theta}\rceil$ невключительно, а остальные занулили. Различных элементов среди $\pi_{\theta,l}(x)$ не больше $2^{\lceil L/\theta\rceil} \leq 2^{L/\theta+1} = 2^{2c\log n/\theta+1} = 2n^{2c/\theta}$. Взяв достаточно большое θ можно уменьшить это значение до сколь угодно малого полинома. Теперь пусть переменная $x_{i,j,t}$ принимает значение 1, если позиции ненулевых бит t в точности совпадают с позициями ненулевых бит в $A_i[j]$. Тогда остается проверить, что $\prod\limits_{i=1}^{k}\prod\limits_{l=1}^{\theta} x_{i,j_i,\pi_{\theta,l}(t_i)} = 1$.

Переменные. Введем $s(n) = O\left(n^{1+2c/\theta}\right)$ переменных:

$$X = \{x_{i,j,t} \colon i \in [k], j \in [n], t \in \{\pi_{\theta,l}(q) :\ 0 \leq l \leq \theta-1, q \in [n^{-c}, n^{c}]\}\}.$$

Отображение $\phi_{s(n)}(I)$ устанавливает следующие значения переменным:

$x_{i,j,t} = 1$, если на позициях, на которых находятся ненулевые биты t, в $A_i[j]$ находятся тоже ненулевые биты; $x_{i,j,t} = 0$, иначе.

Сложность. Сложность представления составляет

$$O\left(n^{1+2c/\theta}\log n\right) = O\left(n^{1+3c/\theta}\right),$$

поскольку для вычисления значения каждой переменной достаточно просмотреть $O(\log n)$ бит числа $A_i[j]$.

Полином.

$$P_{s(n)}(X) = \sum_{\substack{t_1+\ldots+t_k=0,\\ -n^c\leq t_i\leq n^c,\\ j_1,\ldots,j_k\in[n]}} \prod_{i=1}^{k}\prod_{l=0}^{\theta-1} x_{i,j_i,\pi_{\theta,l}(t_i)}.$$

Степень. Степень полинома составляет $k\theta$.

***H*-Изоморфизм подграфа с константным *H*.** Дан граф $G(V,E)$ с $|V| = n, |E| = m$, необходимо определить, содержит ли он подграф, изоморфный фиксированному графу H.

Идея. По определению граф H изоморфен подграфу G, если существует инъ-

ективное отображение $\phi : V(H) \to V(G)$, такое что если $uv \in E(H)$, то $\phi(u)\phi(v) \in E(G)$. Тогда достаточно перебрать все возможные назначения вершин G вершинам H, такие что для различных вершин H назначены различные вершины G, и проверить, что для каждого ребра в H есть ребро в G между соответствующими назначеными вершинами (рёбер и вершин H константное число, поэтому достаточно будет полинома константной степени). Наивная проверка потребовала бы $O(n^2)$ переменных (по одной на каждую пару вершин). Вместо этого для каждого ребра H можно угадывать номер соответствующего ребра в G и поблочно проверять (то есть сравнивать $\pi(\theta, l)(v)$), что инцидентные вершины в точности совпадают с назначеными вершинами.

Переменные. Введем $s(n) = O\left(mn^{2c/\theta}\right)$ переменных:

$$X = \{x_{e,i,v} \colon e \in [m], i \in \{1, 2\}, v \in \{\pi_{\theta,l}(q) : \ 0 \leq l \leq \theta - 1, q \in [n]\}\}.$$

Для каждого ребра графа G укажем, какая его вершина является первой, а какая второй. Отображение $\phi_{s(n)}(I)$ устанавливает следующие значения переменным:

$x_{e,i,v} = 1$, если на позициях, на которых находятся ненулевые биты v, в номере i-той вершины ребра с номером e находятся тоже ненулевые биты; $x_{e,i,v} = 0$, иначе.

Сложность. Сложность представления составляет

$$O\left(mn^{2c/\theta} \log n\right) = O\left(mn^{3c/\theta}\right) = O\left((n+m)^{1+3c/\theta}\right),$$

поскольку для вычисления значения каждой переменной достаточно просмотреть $O(\log n)$ бит.

Полином. Пусть $|V(H)| = h$, $|E(H)| = r$, $c(u, v)$ для $u, v \in V(H)$ обозначает номер ребра u, v в графе H. Для всех наборов различных вершин $v_1, \ldots, v_h \in [n]$, для всех наборов номеров различных рёбер

$e_1, \ldots, e_r \in [m]$, добавим моном

$$\prod_{(a,b)\in E(H)} \sum_{(j_1,j_2)\in\{(1,2),(2,1)\}} \prod_{l=0}^{\theta-1} x_{e_{c(a,b)},j_1,\pi(\theta,l)(v_a)} x_{e_{c(a,b)},j_2,\pi(\theta,l)(v_b)}.$$

Степень. Степень полинома составляет $2|E(H)|\theta$.

□

Теорема 2. *Пусть существует хотя бы одно из следующих сведений ($q > 1$):*

- *n^q-SETH-сведение к задаче **k-Сумма с константным k**;*
- *$(n+m)^q$-SETH-сведение к задаче **H-Изоморфизм подграфа с константным H** (k^q-SETH-сведение к задаче с параметром $k = n + m$);*

Тогда выполняется как минимум одно из следующих утверждений:

- *Класс E^{NP} не вычисляется семейством булевых параллельно-последовательных схем размера $O(n)$;*
- *Для некоторой $\gamma > 1$ существует явный полином константной степени, который не вычисляется арифметическими схемами размера n^γ.*

Доказательство. Теорема непосредственно следует из Теоремы 4 и Леммы 4.
□

Заключение

Данная работа является продолжением [6], она показывает барьеры SETH-сводимости для более широкого круга как NP-трудных, так и полиномиальных задач: ***Наименьшая общая надстрока***, ***Наименьшее общее разбиение строк***, ***Наименьшее общее разбиение строк с константным алфавитом***, ***k-Мотив графа***, ***k-Сумма с константным k***, ***H-Изоморфизм графа с константным H***. В качестве заключения приведем некоторые открытые проблемы, которые возникают в области тонких оценок и связаны с темой работы.

Первая проблема – найти полиномиальное представление сложности $2^{n/\theta}$ для любого $\theta > 1$ для ***Задачи об ударяющем множества***. В [11] показано, что ***Задача об ударяющем множества*** 2^n-SETH-трудная, значит существование такого полиномиального представления автоматически давало бы нижние оценки на схемную сложность. Вторая, не менее важная проблема, найти барьеры для несводимости в предположении других гипотез: 3-SUM, APSP, SCH и др. Возможно ли использовать для этого похожую машинерию, которая была развита в [14] и [6]?

Список литературы

[1] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. Annals of Mathematics, 160:781–793, 2004.

[2] Sanjeev Arora and Boaz Barak. Computational complexity: a modern approach. Cambridge University Press, 2009.

[3] Noga Alon. Explicit construction of exponential sized families of k-independent sets. Discrete Mathematics, 58(2):191–193, 1986.

[4] Noga Alon and Joel H. Spencer. The Probabilistic Method, Third Edition. Wiley, 2008.

[5] Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. Journal of the ACM, 42(4):844–856, 1995.

[6] Tatiana Belova, Alexander Golovnev, Alexander S. Kulikov, Ivan Mihajlin, and Denil Sharipov. Polynomial formulations as a barrier for reduction-based hardness proofs. In SODA, pages 3245–3281. SIAM, 2023.

[7] Avrim Blum, Tao Jiang, Ming Li, John Tromp, and Mihalis Yannakakis. Linear approx- imation of shortest superstrings. In STOC 1991, pages 328–336. ACM, 1991.

[8] Sebastian Bocker, Florian Rasche, and Tamara Steijger. Annotating Fragmentation Patterns. In Proc. of the 9th International Workshop Algorithms in Bioinformatics (WABI), volume 5724 of LNCS, pages 13–24. Springer, 2009.

[9] Bonnet, E., Sikora, F. (2017). The graph motif problem parameterized by the structure of the input graph. Discrete Applied Mathematics, 231, 78-94.

[10] Walter Baur and Volker Strassen. The complexity of partial derivatives. Theoretical computer science, 22(3):317–330, 1983.

[11] Marek Cygan, Holger Dell, Daniel Lokshtanov, Daniel Marx, Jesper Nederlof, Yoshio Okamoto, Ramamohan Paturi, Saket Saurabh, and Magnus Wahlstrom.

On problems as hard as CNF-SAT. ACM Transactions on Algorithms, 12(3):41:1–41:24, 2016.

[12] Cygan, M., Kulikov, A. S., Mihajlin, I., Nikolaev, M., Reznikov, G. (2021). Minimum common string partition: Exact algorithms. In 29th Annual European Symposium on Algorithms (ESA 2021). Schloss-Dagstuhl-Leibniz Zentrum fur Informatik.

[13] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Daniel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. Parameterized algorithms, volume 5. Springer, 2015.

[14] Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility. In ITCS 2016, pages 261–270. ACM, 2016.

[15] Jeff Erickson. Bounds for linear satisfiability problems. Chicago Journal of Theoretical Computer Science, page 8, 1999.

[16] Michael R. Fellows, Guillaume Fertin, Danny Hermelin, and Stéphane Vialette. Upper and lower bounds for finding connected motifs in vertex-colored graphs. J. Comput. Syst. Sci., 77(4):799–811, 2011.

[17] John Gallant. String compression algorithms. PhD thesis, Princeton, 1982.

[18] Golovnev, Alexander, Alexander S. Kulikov, and Ivan Mihajlin. Solving SCS for bounded length strings in fewer than 2^n steps. Information Processing Letters 114.8 (2014): 421-425.

[19] Golovnev, A., Kulikov, A. S., Logunov, A., Mihajlin, I., Nikolaev, M. (2019). Collapsing Superstring Conjecture. In Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2019). Schloss-Dagstuhl-Leibniz Zentrum fur Informatik.

[20] Golovnev, Alexander, Alexander S. Kulikov, and Ivan Mihajlin. Solving 3-superstring in $3^{n/3}$ time. Mathematical Foundations of Computer Science 2013: 38th International Symposium, MFCS 2013, Klosterneuburg, Austria, August 26-30, 2013. Proceedings 38. Springer Berlin Heidelberg, 2013.

[21] John Gallant, David Maier, and James A. Storer. On finding minimal length superstrings. J. Comput. Syst. Sci., 20(1):50–58, 1980.

[22] Avraham Goldstein, Petr Kolman, and Jie Zheng. Minimum common string partition problem: Hardness and approximations. Electron. J. Comb., 12, 2005.

[23] Anka Gajentaan and Mark H. Overmars. On a class of $O(n^2)$ problems in computational geometry. Computational geometry, 5(3):165–185, 1995.

[24] Allan Gronlund and Seth Pettie. Threesomes, degenerates, and love triangles. Journal of the ACM, 65(4):22:1–22:25, 2018.

[25] Russell Impagliazzo and Ramamohan Paturi. The complexity of k-SAT. In CCC 1999, pages 237–240. IEEE, 1999.

[26] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? In FOCS 1998, pages 653–653. IEEE, 1998.

[27] Hendrik W. Lenstra, Jr. and Carl Pomerance. Primality testing with gaussian periods. Journal of the European Mathematical Society, 21(4):1229–1269, 2019.

[28] Vincent Lacroix, Cristina G. Fernandes, and Marie-France Sagot. Motif search in graphs: application to metabolic networks. IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB), 3(4):360–368, 2006.

[29] Jiatu Li and Tianqi Yang. 3.1n - o(n) circuit lower bounds for explicit functions. In STOC 2022, pages 1180–1193. ACM, 2022.

[30] Patrascu, Mihai, and Ryan Williams. On the possibility of faster SAT algorithms. In Proceedings of the twenty-first annual ACM-SIAM symposium

on Discrete Algorithms, pp. 1065-1075. Society for Industrial and Applied Mathematics, 2010.

[31] Ron Y. Pinter, Hadas Shachnai, and Meirav Zehavi. Deterministic parameterized algorithms for the graph motif problem. In Proc. of Mathematical Foundations of Computer Science MFCS 2014, volume 8635 of LNCS, pages 589–600. Springer, 2014.

[32] Pavel A. Pevzner, Haixu Tang, and Michael S. Waterman. An eulerian path approach to DNA fragment assembly. Proc. Natl. Acad. Sci. U.S.A., 98(17):9748–9753, 2001. SODA 2010, pages 1065–1075. SIAM, 2010.

[33] James A. Storer. Data compression: methods and theory. Computer Science Press, Inc., 1987.

[34] Volker Strassen. Die berechnungskomplexit at von elementarsymmetrischen funktionen und von interpolationskoeffizienten. Numerische Mathematik, 20(3):238–251, 1973.

[35] Leslie G. Valiant. Graph-theoretic arguments in low-level complexity. In MFCS 1977, pages 162–176, 1977.

[36] Virginia Vassilevska Williams. Hardness of easy problems: Basing hardness on popular conjectures such as the strong exponential time hypothesis (invited talk). In IPEC 2015, pages 17–29. Schloss Dagstuhl, 2015.

[37] Virginia Vassilevska Williams. On some fine-grained questions in algorithms and com- plexity. In ICM 2018, 2018.

[38] Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. In STOC 2010, pages 645–654. IEEE, 2010.

[39] Michael S. Waterman. Introduction to computational biology: maps, sequences and genomes. CRC Press, 1995.

[40] Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. Theoretical Computer Science, 348(2-3):357–365, 2005.

[41] Ryan Williams. Finding paths of length k in $O^*(2^k)$ time. Information Processing Letters, 109(6):315–318, 2009.