Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет
«Высшая школа экономики»
Факультет Санкт-Петербургская школа физико-математических и
компьютерных наук
Департамент информатики

Основная профессиональная образовательная программа
«Прикладная математика и информатика»

Андреев Никита Валерьевич

Вычислительная сложность задачи парного доминирования в графах

Бакалаврская работа

Допущена к защите.
Зав. кафедрой:
д. ф.-м. н., профессор Омельченко А. В.

Научный руководитель:
к.ф.-м.н. И.А. Близнец

Рецензент:
м.н.с. Д.Г. Сагунов

Санкт-Петербург
2021

Оглавление

Введение **6**

1. Обзор литературы **12**
1.1. PDS: обзор литературы 12
1.2. DS: обзор литературы 13
1.2.1. Параметризация размером ответа 14
1.2.2. Параметризация древесной шириной 16
1.2.3. Параметризация расстоянием 17
1.3. Выводы 18

2. Точный экспоненциальный алгоритм **19**
2.1. Общее описание алгоритма 19
2.2. Предварительные факты 20
2.3. Алгоритм для нахождения минимального покрывающего паросочетания 21
2.4. Выводы 23

3. Алгоритм, параметризованный древесной шириной **24**
3.1. Динамическое программирование для простых случаев . 24
3.2. Динамическое программирование для join node 26
3.3. Выводы 28

4. Оптимальность алгоритма, параметризованного древесной шириной **29**
4.1. Построение графа по булевой формуле 29
4.2. Выполнимость влечёт наличие PDS 31
4.3. Наличие PDS влечёт выполнимость 32
4.4. Древесная декомпозиция полученного графа 33
4.5. Выводы 34

5. Параметризация расстоянием **35**
5.1. Общая схема алгоритмов 35
5.2. Решение SC_p(cluster graphs) 36

5.3. Интервальные графы 38
5.3.1. Необходимые определения 38
5.3.2. Решение SC(interval graphs) 39
5.3.3. Решение SC_t(interval graphs) 40
5.4. Графы перестановок 41
5.4.1. Решение SC(permutation graphs) 42
5.4.2. Решение SC_t(permutation graphs) 43
5.5. Потенциальное улучшение времени работы 44
5.6. Выводы 50

6. Заключение **51**

Список литературы **52**

Dominating Set - одна из классических NP-полных задач Computer Science. В ней для данного графа нужно найти такое наименьшее подмножество вершин, что любая из оставшихая вершин имеет соседа из этого множества. Помимо классической формулировки, задача имеет множество вариаций. Одна из таких вариаций - Paired Dominating Set, в которой требуется, чтобы выбранные вершины можно было разбить на пары смежных вершин. В данной работе изучается вычислительная сложность задачи Paired Dominating Set. Были разработаны новые методы и придуманы новые алгоритмы. Среди них есть как точные экспоненциальные, так и параметризованные различными структурными параметрами.

Ключевые слова: Dominating Set, Paired Dominating Set, Total Dominating Set, Точные экспоненциальные алгоритмы, Параметризованные алгоритмы, Древесная ширина, SETH, Distance from Triviality

The Dominating Set is one of the classic NP-complete problems in Computer Science. Given a graph, we need to find the smallest subset of vertices such that any of the remaining vertices has a neighbor from this set. In addition to the classical formulation, the problem has many variations. One such variation is the Paired Dominating Set, which requires that the selected vertices can be split into pairs of adjacent vertices. This paper examines the computational complexity of the Paired Dominating Set problem. New methods have been developed and new algorithms have been invented. Among them there are both exact exponential algorithms and algorithms, parameterized by various structural parameters.

Keywords: Dominating Set, Paired Dominating Set, Total Dominating Set, Exact exponential algorithms, Parametrized algorithms, Treewidth, SETH, Distance from Triviality

Введение

Задача доминирующего множества (dominating set), или сокращённо DS, является одной из классических NP-полных задач, и имеет широкий круг применений от организации компьютерных сетей [1] и оптимизации систем наблюдения [12] до автоматической генерации выжимки текста [27]. Задача DS имеет множество вариаций, таких, как Total Dominating Set, Connected Dominating Set, Independent Dominating Set. Одна из таких вариаций - Paired Dominating Set или PDS является темой изучения данной работы.

Ключевые определения и мотивация

Для графа G будем обозначать $V(G)$ - множество вершин графа, $E(G)$ - множество рёбер. Для $x \in V(G)$ за $N(x)$ обозначим множество соседей x, а за $N[x]$ обозначим $N(x) \cup \{x\}$. Будем говорть $f(n) = \mathcal{O}^*(g(n))$, если существует полином $p(n)$ такой, что для достаточно больших n выполнено $f(n) < p(n) \cdot g(n)$.

Задача DS определяется следующим образом:

Определение 0.1 (Dominating Set)**.** Для заданного графа G требуется найти минимальное множество вершин $D \subset G$ такое, что выполнено $\forall v \in V(G) \setminus D \, \exists w \in D : (v, w) \in E(G)$. В таком случае будем говорить, что вершина w доминирует v. Эквивалентное определение: $\bigcup\limits_{v \in D} N[v] = V(G)$.

Иногда задачу DS формулируют так, что необходимо только найти размер минимального доминирующего множества. В данной работе некоторые алгоритмы используют именно такую формулировку, однако очевидным образом обощаются на случай нахождения самого множества.

Можно представить себе граф как некоторую область, которая должна быть под наблюдением. Для этого в некоторых местах можно разместить датчики, которые будут контролировать прилежащую окресность, а именно все вершины, смежные с вершиной, в которой располо-

жен датчик. Тогда для того, чтобы покрыть всю область наблюдением, расставив при этом минимальное количество датчиков, требуется решить задачу DS.

Для того, чтобы описать мотивацию PDS, давайте предположим, что датчики могут выходить из строя, и, чтобы гарантировать свою работоспособность, они периодически рассылают сигнал соседним датчикам. Тогда можно наложить требование, чтобы у каждого датчика среди соседей был другой датчик. Это позволяет сформулировать задачу Total Dominating Set, или TDS:

Определение 0.2 (Total Dominating Set)**.** Для заданного графа G требуется найти минимальное множество вершин $D \subset G$ такое, что выполнено $\forall v \in V(G)\, \exists w \in D : (v, w) \in E(G)$. Эквивалентное определение: $\bigcup\limits_{v \in D} N(v) = V(G)$.

Развивая идею, можно предположить, что если каждый датчик шлёт сигнал каждому из соседних, может возникнуть ситуация, когда какой-то датчик имеет много содених датчиков и находится под большой нагрузкой. Разумно разбить датчики на пары, обеспечив прикрытие датчиков внутри каждой из пар. Это приводит нас к следующему определению, которое было введено в 1998 [13]:

Определение 0.3 (Paired Dominating Set)**.** Для заданного графа G требуется найти минимальное множество вершин $D \subset G$ такое, что выполнено $\forall v \in V(G) \setminus D\, \exists w \in D : (v, w) \in E(G)$, а также вершины множества D можно разбить на пары смежных, то есть в D есть совершенное паросочетание.

Легко придумать алгоритмы для вышеупомянутых задач, которые бы работали за время $\mathcal{O}^*\left(2^{|V(G)|}\right)$. Для этого просто нужно перебирать все подмножества вершин, и среди подходящих выбрать минимальное по размеру. Интереснее придумать алгоритм, который бы работал за время $\mathcal{O}^*\left(\alpha^{|V(G)|}\right)$, где $\alpha < 2$. Известны экспоненциальные алгоритмы для задачи DS. На данный момент лучшее время ([28]) для задачи DS равно $\mathcal{O}^*\left(1.4969^{|V(G)|}\right)$. Однако до сих пор не было представлено такого алгоритма для задачи PDS.

Параметризованные алгоритмы

Помимо точных экпоненциальных алгоритмов, широко развита область параметризованных алгоритмов.

Определение 0.4 (FPT)**.** Задача является fixed parameter tractable (FPT) с параметром k, если для нее существует алгоритм, решающий ее за время $f(k) \cdot |x|^{\mathcal{O}(1)}$, где f некая функция, не зависящая от $|x|$, а x – вход для задачи. Данный алгоритм и является параметризованным алгоритмом. Равносильно, время равно $\mathcal{O}^*(f(k))$.

В качестве параметра может выступать, например, размер ответа, или другие структурные параметры графа, основные из которых будут представлены в разделе 1. Существует иерархия классов параметризованных задач: $FPT \subset W[1] \subset W[2] \subset ....$ Классу FPT принадлежат задачи, допускающие параметризованные алгоритмы. Предполагается, хоть и не доказано, что для задач из $W[1]$ и больших не существует алгоритмов, работающих за FPT-время. Каждый из этих классов замкнут относительно параметризованной редукции:

Определение 0.5. Параметризованная редукция это алгоритм, преобразующий вход x задачи A с параметром k к входу x' задачи B с параметром k' так, что

- ответ для (x, k) совпадает с ответом на (x', k')
- $k' < g(k)$ для некоторой вычислимой функции g
- время работы алгоритма $f(k) \cdot |x|^{\mathcal{O}(1)}$

$W[k]$-трудной называется такая задача X, что любая задача Y из $W[k]$ допускает параметризованную редукцию из Y в X. Известно, что задача DS является $W[2]$-трудной.

Древесная декомпозиция

Одна из областей изучения вычислительной сложности задач - изучать сложность относительно каких-то фиксированных параметров. В случае графов в качестве параметра можно рассмотреть древесную ширину, или ширину фиксированной древесной декомпозиции:

Определение 0.6. Пусть есть граф G. Древесной декомпозицией называется дерево T, с каждой вершиной t которой ассоциирован некоторый подграф H_t, состоящий из вершин $X_t \subset G$, и которое удовлетворяет следующим условиям:

- $\bigcup X_t = V(G)$
- $\forall (u, v) \in E(G)\ \exists t \in V(T) : u, v \in E(H_t)$
- Для любой $v \in V(G)$ все вершины t, для которых $v \in X_t$ образуют связное множество в T

Шириной разложения называется максимальный размер множеств $|X_t|$ минус один. Древесной шириной графа называется минимальная ширина среди всех древесных разложений.

Расстояние до простых случаев

Другим параметром, интересным для изучения является расстояние до тривиальных случаев (distance from triviality). Пусть есть класс графов A, для которого задача может быть решена за полиномиальное время. Если из нашего графа G можно выбросить минимальное число d вершин так, чтобы получить граф, принадлежать A, будем говорить, что расстояние до A равно d. Можно рассматривать d в качестве параметра, и придумывать эффективные алгоритмы, параметризованные этим расстоянием. Множество вершин S, которое необходимо удалить называется модулятором.

Важно отметить, что говоря про параметризованные алгоритмы, имеется ввиду, что на вход подаётся не только граф, но ещё и сам параметр. Например, в задаче, параметризованной размером ответа, на вход подаётся k, и требуется узнать, есть ли ответ размера не больше (или не меньше в случае задачи максимизации), чем k. При параметризации древесной шириной на вход подаётся древесная декомпозиция, а при параметризации расстоянием - на вход подаётся модулятор.

Нижние оценки и гипотезы

При изучении вычислительной сложности задач интересно не только придумывать эффективные алгоритмы для их решения, но и доказывать оптимальность последних, или получать нижние оценки на их время работы. Однако доказательство нижней оценки в чистом виде представляет из себя крайне сложную задачу и часто сводится к проблеме равенства классов P и NP. Поэтому обычно предполагается верность какой-то известной гипотезы, и из этого выводится результат. Самая известная из таких гипотез утверждает, что P≠NP. В данной работе будет использоваться гипотеза SETH (Strong Exponential Time Hypothesis). Определим числа δ_i как инфимум таких чисел δ, что задача k-SAT может быть решена за время $\mathcal{O}\left(2^{\delta n}\right)$, где n - количество переменных. Ясно, что $\delta_1 < \delta_2 < \delta_3 < ....$

Определение 0.7. Гипотеза ETH (Exponential Time Hypothesis) утверждает, что $\delta_3 > 0$.

Определение 0.8. Гипотеза SETH (Strong Exponential Time Hypothesis) утверждает, что $\lim\limits_{k\to\infty} \delta_k = 1$.

Вторая гипотеза является более сильной, и из неё следует, что задача SAT не может быть решена за время $\mathcal{O}^*\left((2-\varepsilon)^n\right)$.

Цель и задачи

Целью данной работы является изучение вычислительной сложности и нахождение эффективных методов для решения задачи PDS.

Задачи:

- Изучить сложность относительно параметра "размер ответа".
- Придумать экспоненциальный алгоритм со временем $\mathcal{O}^*\left((2-\varepsilon)^n\right)$.
- Изучить сложность задачи относительно различных структурных параметров, по возможности получить эффективные параметризованные алгоритмы и нижние оценки на время их работы.

Достигнутые результаты

- Доказана $W[2]$-трудность для задачи PDS и проанализирована нижняя оценка на время работы алгоритмов.
- Представлен точный экспоненциальный алгоритм для PDS, имеющий время работы $\mathcal{O}^*(1.7159^n)$.
- Представлен алгоритм для PDS, параметризованный шириной k древесного разложения, имеющий время работы $\mathcal{O}^*\left(4^k\right)$
- В предположении гипотезы SETH доказана оптимальность такого алгоритма.
- Представлен алгоритм для PDS, параметризованный расстоянием d до кластерных графов, имеющий время работы $\mathcal{O}^*\left(4^d\right)$.
- Представлены алгоритмы для DS и TDS, параметризованные расстоянием d до интервальных графов и графов перестановок, имеющие время работы $\mathcal{O}^*\left(3^d\right)$ и $\mathcal{O}^*\left(4^d\right)$ соответственно.
- Был проведён анализ времени работы таких алгоритмов в зависимости от размера независимого множества в модуляторе.

Структура работы

В главе 1 представлен обзор существующих работ и результатов в области PDS а также DS. Кроме того, результаты, касающиеся параметризации размером ответа для DS обобщаются на случай PDS.

В главе 2 представлен точный экспоненциальный алгоритм, решающий задачу PDS за время $\mathcal{O}^*(1.7159^n)$.

В главе 3 представлен алгоритм, параметризованный древесной шириной, решающий задачу PDS за время $\mathcal{O}^*\left(4^k\right)$.

В главе 4 доказана оптимальность алгоритма, представленного в главе 3.

В главе 5 представлена общая схема алгоритмов для DS и его вариаций, параметризованных расстоянием до классов графов, а также представлено несколько алгоритмов для задач PDS, DS, TDS.

1. Обзор литературы

1.1. PDS: обзор литературы

В 1998 году Teresa Haynes и Peter Slater [13] ввели задачу Paired Dominating Set. В данной статье были охарактеризованы все тройки чисел $a < b < c$, для которых существуют графы такие, что $\gamma(G) = a$, $\gamma_t(G) = b$, $\gamma_p(G) = c$, где $\gamma(G)$, $\gamma_t(G)$, $\gamma_p(G)$ - размеры минимального DS, TDS и PDS соответственно. Другой результат, более интересный для данной работы, это доказательство NP-полноты задачи PDS. Авторы представили сведение задачи DS к PDS, и воспользовались NP-полнотой задачи DS. Мы опишем данное сведение, и заметим, что оно является параметризованной редукцией. Это будет означать, что PDS является $W[2]$-трудной, так как $W[2]$-трудной является задача DS.

Теорема 1. Задача PDS является $W[2]$-трудной.

Доказательство. (Haynes T., Slater P.) Рассмотрим граф G. Построим по нему граф H следующим образом: рассмотрим четыре копии $V(G)$ - V_1, V_2, V_3 и V_4. Внутри этих копий рёбра будут проведены в соответствии с рёбрами исходного графа. Рассмотрим копии V_1 и V_2. пусть $v \in V_1$ и $w \in V_2$. Пусть v', w' - соответствующие им вершины исходного графа. Проведём между v и w ребро тогда и только тогда, когда либо $v' = w'$, либо между v' и w' проведено ребро. Сделаем тоже самое с копиями V_3 и V_4. После этого добавим все рёбра между одинаковыми вершинами копий V_2 и V_3.

Тогда в графе G есть DS размера не более k тогда и только тогда, когда в графе H есть PDS размера не более $2k$. Имея DS в G размера k, взяв соответствующие вершины из копий V_2 и V_3, легко получить PDS размера $2k$. Обратно, имея PDS размера $2k$, рассмотрим его ограничение R на $V_1 \cup V_2$. Из симметрии, можно считать, что $R < k$. Взяв соответствующие вершины в G, получим DS размера не более k. □

В 2010 году L. Chen, C. Lu, и Z. Zeng [6] показали NP-полноту задачи PDS, ограниченной только на двудольные графы, хордальные графы

или расщепляемые (split) графы. Там же были представлены полиномиальные алгоритмы для блоковых и интервальных графов. Известно множество других полиномиальных алгоритмов для других классов, более полный обзор представлен в таблице:

Класс графов	Сложность	Год	Статья
Деревья	$\mathcal{O}(n)$	2003	[11]
Взвешенные деревья	$\mathcal{O}(n)$	2009	[4]
Inflated Trees	$\mathcal{O}(n)$	2004	[15]
Строго хордальные графы	$\mathcal{O}(n+m)$	2009	[5]
Графы перестановок	$\mathcal{O}(n)$	2009	[16]
Выпуклые двудольные	$\mathcal{O}(n+m)$	2010	[21]
Блоковые графы	$\mathcal{O}(n+m)$	2010	[6]
Интервальные графы	$\mathcal{O}(n+m)$	2010	[6]
Графы дуг окружности	$\mathcal{O}(n+m)$	2015	[17]
Strongly Orderable Graphs	$\mathcal{O}(n+m)$	2019	[23]

1.2. DS: обзор литературы

Поскольку задача PDS является вариацией задачи DS, важно изучить работы, касающиеся DS. За все время изучения сложности DS было предложено множество точных экспоненциальных алгоритмов. В данной таблице представлены последние результаты:

Сложность	Год	Статья
$\mathcal{O}(1.5063^n)$	2008	[25]
$\mathcal{O}(1.5048^n)$	2009	[29]
$\mathcal{O}(1.4969^n)$	2011	[28]

Кроме задачи нахождения наименьшего по размеру доминирующего множества существует более сложная задача перечисления всех минимальных по включению доминирующих множеств. Назовём эту задачу Dom-Enum. В 2008 F. V. Fomin, F. Grandoni, A. V. Pyatkin и A.

A. Stepanov [7] представили алгоритм, решающий Dom-Enum за время $\mathcal{O}^*(1.7159^n)$. Известны также результаты, решающие данную задачу за лучшее время для частных случаев графов:

Класс графов	Сложность	Год	Статья
Общий случай	$\mathcal{O}^*(1.7159^n)$	2008	[7]
Хордальные	$\mathcal{O}^*(1.5048^n)$	2019	[8]
Расщепляемые, индифферентные	$\mathcal{O}^*(1.4656^n)$	2013	[19]
Кографы	$\mathcal{O}^*(1.5704^n)$	2013	[19]

1.2.1. Параметризация размером ответа

Так как задача DS является $W[2]$-трудной относительно размера ответа k, считается, что для неё не существует алгоритма со временем работы $\mathcal{O}^*(f(k))$. Однако, простой перебор подмножеств размера k даст время, равное $\mathcal{O}(n^{k+\mathcal{O}(1)})$. В 2010 году Mihai Patra и Ryan Williams [24], в предположении SETH, представили доказательство следующего утверждения:

Теорема 2. ([24]) Для любого $k \geq 3$ и $\varepsilon > 0$ не существует алгоритма для DS, параметризованного размером ответа k, работающего за время $\mathcal{O}\left(n^{k(1-\varepsilon)}\right)$.

Результат легко обобщается на случай PDS. Для этого приведём доказательство используемой леммы, лишь немного модифицируя построение:

Лемма 3. ([24]) Пусть существует $k \geq 3$ такое, что DS, параметризованная размером ответа k может быть решена за время $\mathcal{O}\left(n^{k(1-\varepsilon)}\right)$. Тогда задача SAT может быть решена за время $\mathcal{O}(m + k2^{n/k})^{k(1-\varepsilon)} = \mathcal{O}\left(\max\left(m^{k(1-\varepsilon)}, 2^{n(1-\varepsilon)}\right)\right)$.

Доказательство. Пусть задана булева формула F, содержащая n переменных и m клозов. Построим по ней граф G, в котором будет DS или PDS (в случае чётного k) размера k тогда и только тогда, когда F выполнима. Для этого разобьём переменные на k групп размера $\frac{n}{k}$,

для каждого означивания каждой группы заведём вершину v. Получим $k2^{n/k}$ вершин. Для каждого клоза заведём вершину w, и соединим её со всеми вершинами v, соответствующими означиваниям, которые выполняют данный клоз. Для каждой из k групп заведём фиктивную вершину u, и соединим её со всеми вершинами v из данной группы. После этого обьединим все вершины v в клику.

Тогда, если у формулы F есть выполняющее означивание, взяв в каждой группе соответствующую вершину v, получим доминирующее множество размера k. При чётном k можно разбить взятые вершины на пары, так как все вершины v образуют клику, и получить парное доминирующее множество.

Обратно, имея (парное) доминирующее множество размера k в построенном графе, в каждой группе будет взято ровно по одной вершине v, поскольку иначе какая-то из вершин u была бы не продоминирована. Тогда взятые вершины зададут выполняющее F означивание, поскольку продоминированы все вершины w, соответствующие клозам.

Всего в графе $m + k + k2^{n/k}$ вершин, имея алгоритм, решающий DS за время $\mathcal{O}\left(n^{k(1-\varepsilon)}\right)$, мы бы получили алгоритм, решающий SAT за время $\mathcal{O}(m+k2^{n/k})^{k(1-\varepsilon)}$. Поскольку $k = \mathcal{O}(1)$ фиксировано, время равно $\mathcal{O}\left(\max\left(m^{k(1-\varepsilon)}, 2^{n(1-\varepsilon)}\right)\right)$. □

Теорема 4. Пусть SETH верна. Тогда не существует такого k и $\varepsilon > 0$, что задача PDS, параметризованная размером ответа k может быть решена за время $\mathcal{O}\left(n^{k(1-\varepsilon)}\right)$.

Доказательство. Пусть есть такие k и ε, что можно за время $\mathcal{O}\left(n^{k(1-\varepsilon)}\right)$ ответить есть ли в заданном графе PDS размера не более k. Возьмём такое q, что задача q-SAT не может быть решена за время $\mathcal{O}\left(2^{n(1-\varepsilon)}\right)$. Для любой формулы F - инстанса q-SAT, количество клозов $m \leq (2n)^q$. Тогда $m^{k(1-\varepsilon)} = o(2^{n(1-\varepsilon)})$, и по лемме 3, можно вычислить выполнимость F за время $\mathcal{O}\left(2^{n(1-\varepsilon)}\right)$. □

Помимо нижней оценки, в работе был представлен алгоритм для DS, который работает за время $\mathcal{O}(n^{k+o(1)})$ и использует следующее утверждение:

Теорема 5. ([24]) Пронумеруем подмножества S_i размера k. Определим матрицу M из нулей и единиц размера $\binom{n}{k/2} \times \binom{n}{k/2}$ так, что $M[i,j] = 0$ если и только если $S_i \cup S_j$ - доминирующее множество. Такую матрицу можно вычислить за время $\mathcal{O}(n^{k+o(1)})$ при $k > 7$.

Имея такую матрицу, можно, перебрав все пары S_i, S_j проверить существование доминирующего множества размера не более k за $\mathcal{O}(n^k)$. Очевидным образом проверяя наличие совершенного парсосчетания в $S_i \cup S_j$ за время $k^{\mathcal{O}(1)}$ можно также проверить существование требуемого PDS, либо найти минимальное такое.

Теорема 6. Для заданного k и графа G можно найти наименьшее PDS размера не более k, либо сказать, что его не существует за время $\mathcal{O}\left(n^k \left(n^{o(1)} + k^{\mathcal{O}(1)}\right)\right)$.

1.2.2. Параметризация древесной шириной

При построении алгоритмов, параметризованных древесной шириной, удобно считать что древесная декомпозиция имеет простой вид. В [22] было описано, как, имея древесную декомпозицию, можно привести её к более удобному для рассмотрения виду, а именно:

Определение 1.1. Древесная декомпозиция называется хорошей, если она имеет только вершины определённого вида:

- Leaf node - листовые вершины, для которых $X_t = \emptyset$
- Introduce vertex node - вершина, имеющая вид $H_t = H_{t'} \cup \{v\}$ для ребенка t', без добавления рёбер между $X_{t'}$ и v.
- Introduce edge node - вершина, имеющая вид $X_t = X_{t'}$ для ребенка t', и имеющая ребро $e \in E(H_t)$, отсутствующее в $E(H_{t'})$.
- Forget node - вершина, имеющая вид $X_t = X_{t'} \setminus \{v\}$ для ребенка t', без добавления или удаления рёбер, не инцидентных v.
- Join node - вершина, имеющая ровно двух детей t_1, t_2, для которой $H_t = H_{t_1} = H_{t_2}$.

Теорема 7 (Глава 7 из [22]). Имея древесную декомпозицию T ширины k графа G можно за время $\mathcal{O}\left(k^2 \cdot \max(|V(T)|, |V(G)|)\right)$ получить хорошую древесную декомпозицию, имеющую ширину k и количество вершин $\mathcal{O}(k \cdot |V(G)|)$.

Известен алгоритм для DS, параметризованный древесной шириной:

Теорема 8 (Секция 7.3.2 в [22]). Существует алгоритм, решающий задачу DS за время $\mathcal{O}^*\left(3^k\right)$ для заданного графа и его древесного разложения ширины k.

В 2018 Daniel Lokshtanov, Daniel Marx и Saket Saurabh [18] представили доказательство оптимальности такого алгоритма в предположении SETH:

Теорема 9 ([18]). Предположим, что гипотеза SETH верна. Пусть дан граф G, и его древесная декомпозиция ширины k. Для любого $\varepsilon > 0$ не существует алгоритма, решающего задачу DS за время $\mathcal{O}^*\left((3-\varepsilon)^k\right)$.

1.2.3. Параметризация расстоянием

В 2018 году Jiong Guo, Falk Huffner, и Rolf Niedermeier представили алгоритм для DS, параметризованный расстоянием до кластерных графов (кластрные графы - несвязное обьединение клик):

Теорема 10 ([10]). Для заданного графа G и модулятора $S \subset V(G)$ такого, что $G \setminus S$ - кластерный граф, существует алгоритм, рашающий задачу DS за время $\mathcal{O}^*\left(3^{|S|}\right)$.

При этом также была дана простая нижняя оценка, равная $\mathcal{O}^*\left(2^{|S|}\right)$ в предположении того, что Set-Cover не может быть решена за время $\mathcal{O}^*\left((2-\varepsilon)^{|U|}\right)$, где U - универсум. Вопрос точности этой оценки остаётся открытым.

В 2019 году Neeldhara Misra и Piyush Rathi [20] доказали $W[2]$-трудность задачи DS, параметризованной расстоянием до ко-двудольных графов, в то время как на таких графах задача DS решается тривиальным полиномиальным алгоритмом.

1.3. Выводы

- Существует множество полиномиальных алгоритмов для решения PDS для частных случаев графов, но отсутствует точный экспоненциальный алгоритм.
- Задача DS изучена гораздо более широко, чем PDS, что позволяет обобщить на случай PDS или опровергнуть для него существующие факты, касающиеся DS.
- Параметризация расстоянием изучена только для простейших классов графов, в то время как задача DS может быть решена за полиномиальное время для множества нетривиальных классов графов.

2. Точный экспоненциальный алгоритм

Пусть дан алгоритм, решающий задачу Dom-Enum за время $t(n)$. В данном разделе представлен алгоритм для решения PDS, работающий за время $\mathcal{O}^*(t(n))$.

2.1. Общее описание алгоритма

Пусть D' - минимальное парное доминирующее множество. Ясно, что существует такое минимальное по включению множество $D \subset D'$, которое является доминирующим. Поэтому корректен следующий алгоритм для нахождения минимального PDS:

Algorithm 1: Точный экспоненциальный алгоритм для PDS

Answer = $V(G)$;
for D - *минимальное по включению DS* **do**
    1. $D' \supset D$ - наименьшее надмножество, содержащее совершенное паросочетание;
    2. Answer = min(Answer, D');
end
return *Answer*;

Теорема 11. Пусть алгоритм, решающий Dom-Enum работает за время $t(n)$. Тогда алгоритм 1 работает за время $\mathcal{O}^*(t(n))$.

Доказательство. Достаточно решить пункт 1 алгоритма за полиномиальное время. Доказательство сводится к следующей теореме. □

Теорема 12. Пусть задан граф G, и подмножество вершин $D \subset V(G)$, и требуется найти минимальное надмножество $D' \supset D$, содержащее совершенное паросочетание. Данную задачу можно решить за полиномиальное время.

Оставшаяся часть данного раздела будет посвящена доказательству этой теоремы. Данную задачу можно переформулировать следующим

образом: требуется найти наименьшее паросочетание, покрывающее заданное множество вершин D. Алгоритм, решающий данную задачу состоит из двух этапов, и использует идею увеличивающих путей.

Замечание. Если потребовать от D' отсутствие изолированных вершин или связность, то такая задача окажется NP-полной, так как к ней легко сводится Set Cover, взяв D без рёбер внутри в качестве универсума, и $V(G) \setminus D$ образующие клику, в качестве покрывающих множеств. Поэтому описанный способ не работает для решения Total DS или Connected DS.

2.2. Предварительные факты

Определение 2.1. Чердующимся путем в паросочетании M называется путь, в котором чередуются ребра, принадлежащие M. Если при этом крайние вершины пути не покрыты ребрами из M, путь называется увеличивающим.

Теорема 13 (Claude Berge, 1957)**.** Паросочетание M является наибольшим тогда и только тогда, когда для него не существует увеличивающей цепи.

Приведём доказательство, так как в нём используются общеизвестные но важные идеи, касающиеся увеличивающих путей:

Доказательство.
1. Пусть в графе есть увеличивающий путь, состоящий из рёбер $e_1, e_2, ..., e_{2k+1}$. Все рёбра с четными индексами принадлежат паросочетанию: $e_2, e_4, ..., e_{2k} \in M$. Рассмотрим паросочетание $M' = M \setminus \{e_2, e_4, ..., e_{2k}\} \cup \{e_1, e_3, ..., e_{2k+1}\}$, полученное инвертированием рёбер на данном пути. Тогда M' - паросочетание, так как по условию крайние вершины не принадлежали M, причём $|M'| = |M|+1$, что говорит о том, что M не было максимальным.
2. Пусть есть паросочетание M, не являющееся максимальным. Рассмотрим максимальное паросочетание M'. Рассмотрим симметрическую

разность рёбер $X = M\Delta M' = (M \cup M') \setminus (M \cap M')$. Лего видеть, что каждой вершине инцидентно не более двух ребер из X, поэтому X состоит из чередующихся циклов и путей. Поскольку $|M'| > |M|$, будет существовать путь нечетной длины, состоящий из рёбер $e_1, e_2, ..., e_{2k+1} \in X$, такой, что $e_2, e_4, ..., e_{2k} \in M \setminus M'$, а $e_1, e_3, ..., e_{2k+1} \in M' \setminus M$. Такой путь и будет увеличивающим. □

Теорема 14 (Jack Edmonds, 1961)**.** Для заданного паросочетания M, и заданной вершины v можно за полиномиальное время найти увеличивающий путь, который начинается в v, либо сказать, что его не существует.

Последние две теоремы используются в алгоритмах для поиска максимального паросочетания (Blossom algorithm by Jack Edmond, 1961).

2.3. Алгоритм для нахождения минимального покрывающего паросочетания

Рассмотрим паросочетания M, покрывающие D, то есть такие что для любой вершины $v \in D$, она инцидентна какому-то ребру из M. Наша цель минимизировать размер M, что равносильно минимизации количества вершин $w \in V(G) \setminus D$, покрытых M. Ясно, что далее можно считать, что в $G \setminus D$ нет рёбер, так как такие ребра нет смысла брать в итоговое паросочетание.

Нахождение покрывающего паросочетания

Пусть требуется найти любое паросочетание M, покрывающее D. Пусть такое паросочтание существует, обозначим его за M_0. Возьмём для начала максимальное паросочетание M в G. Рассмотрим $X = M\Delta M_0$. Пусть M не покрывает какую-то вершину $v \in D$. Тогда в X есть максимальный чередующийся путь $v_0, v_1..., v_k : (v_i, v_{i+1}) \in X$, имеющий начало в $v_0 = v$, поскольку v покрыто M_0. Во-первых этот путь имеет чётную длину, так как иначе он был бы увеличивающим путём для M. Значит $(v_{k-1}, v_k) \in M \setminus M_0$. Во-вторых этот путь имеет конец вне D ($v_k \notin D$),

так как последняя вершина не может покрываться M_0 в силу того что путь максимальный. Тогда рассмотрим паросочетание M', полученное инвертированием рёбер на этом пути. Легко понять, что $|M'|= |M|$, и количество покрытых вершин из D увеличилось на 1. Поиск вышеописанного пути можно свести к поиску увеличивающего пути следующим образом: добавим в граф фиктивную вершину w, и соединим с ней рёбрами все вершины из $V(G) \setminus D$. Тогда v_0, $v_1 ..., v_k$, w - увеличивающий путь для M.

Algorithm 2: Поиск паросочетания, покрывающего D

```
M = максимальное паросочетание в G;
while M не покрывает D do
    P - увеличивающий путь в G ∪ {w}, который начинается в
     непокрытой вершине из D;
    if P существует then
        инвертировать рёбра на пути P, убрать ребро
         инцидентное w;
    else
        return не существует требуемого паросочетания;
    end
end
return M;
```

Уменьшение покрывающего паросочетания

Пусть на данном этапе алгоритма мы имеем паросочетание M, покрывающее D. Рассмотрим его ограничение на ребра, имеющие границы в D, то есть $M_D = M \cap E(D)$. Если M_D максимально в D, то M минимально, так как тогда минимально количество вершин из $V(G) \setminus D$, покрытых M. Если же M_D не является максимальным, то для него существует увеличивающий путь. Возьмем крайние вершины этого пути v_1, $v_2 \in D$, они не инцидентны никаким ребрам из M_D. Но тогда они инцидентны некоторым ребрам из $M \setminus M_D$, поскольку M покрывает D. То есть существуют вершины w_1, $w_2 \in V(G) \setminus D$, такие, что (v_1, w_1), $(v_2, w_2) \in M$. Выбросим рёбра (v_1, w_1), (v_2, w_2) из паросочетания

и инвертируем рёбра на этом пути. Получим паросочетание M', которое по-прежнему покрывает D, и которое имеет больше ребер в $E(D)$. Таким образом, за конечное число шагов можно будет получить искомое паросочетание.

Algorithm 3: Минимизация паросочетания, покрывающего D

$M =$ паросочетание в G, покрывающее D;
$M_D = M \cap E(D)$;
while M_D *не максимально в* D **do**
  P - увеличивающий путь для M_D в D;
  $e_1, e_2 \in M$ - рёбра, инцидентные крайним вершинам в P;
  инвертировать рёбра на пути P, убрать рёбра e_1, e_2;
end
return M

2.4. Выводы

Был предложен полиномиальный алгоритм, для нахождения минимального паросочетания, покрывающего заданное подмножество вершин. С помощью него был получен алгоритм для задачи PDS, работающий за время $\mathcal{O}^*(t(n))$, где $t(n)$ - время алгоритма, решающего задачу Dom-Enum. Лучший алгоритм на текущий момент [7], решающий задачу Dom-Enum, работает за время $\mathcal{O}^*(1.7159^n)$, поэтому PDS также может быть решена за время $\mathcal{O}^*(1.7159^n)$.

3. Алгоритм, параметризованный древесной шириной

В секции 1.2.2 было введено понятие древесной декомпозиции, и описано, какой вид имеет хорошая декомпозиция. А именно, хорошая древесная декомпозиция представляет собой дерево, в которой каждая вершина имеет вид leaf node, introduce vertex node, introduce edge node, forget node или join node. Имея такую декомпозицию, можно построить эффективный алгоритм для задачи PDS, применив идею динамического программирования. Для этого подвесим дерево за любую вершину, и будем идти от листьев к корню, пересчитывая значение интересующей функции в каждой вершине через её детей.

3.1. Динамическое программирование для простых случаев

Введём в рассмотрение функции $f : X_t \to \{1, 0, \hat{1}, \hat{0}\}$ для каждой вершины дерева X_t, назовем их покрасками, а значения функции - цветами. Обозначим за G_t обьединение всех $X_{t'}$ в поддереве t. Будем искать частичные решения в G_t, соответствующие данной покраске, а именно определим функционал:

Определение 3.1. Для каждой вершины t и покраске f на ней определим $c[t, f]$ как размер минимального множества $D \subset G_t$, такого, что

- $D \cap X_t = f^{-1}\{1, \hat{1}\}$ (среди X_t мы берём в ответ ровно вершины, покрашенные в 1 или $\hat{1}$)
- $G_t \setminus f^{-1}\left(\hat{0}\right)$ продоминировано D (вершины, покрашенные в $\hat{0}$ могут не быть продоминированы)
- $D \setminus f^{-1}\left(\hat{1}\right)$ содержит совершенное паросочетание (вершины, покрашенные в $\hat{1}$ могут не иметь пары среди остальных вершин из D)

Будем говорить, что множество D выполняет условия для покраски f.

Достаточно показать, как можно пересчитывать значение $c[t,\, f]$ через детей t. Рассмотрим все случаи типа вершины t.

Leaf node

Для листовых вершины мы имеем $X_t = \emptyset$, поэтому для единственной пустой покраски имеем $c[t,\, \emptyset] = 0$.

Introduce vertex node

Вершина имеет вид $X_t = X_{t'} \cup \{v\}$ для ребенка t'. Поскольку вершина v изолирована в G_t, верно следующее:

$$c[t,\, f] = \begin{cases} +\infty, & f(v) \in \{0, 1\} \\ c[t',\, f|_{X_{t'}}], & f(v) = \hat{0} \\ 1 + c[t',\, f|_{X_{t'}}], & f(v) = \hat{1} \end{cases}$$

Introduce edge node

Вершина имеет вид $X_t = X_{t'}$, где в графе H_t имеется новое реборо uv, $u,\, v \in X_t$. Это ребро может влиять на доминирование, только если одна вершина покрашена в 1 или $\hat{1}$, а вторая в 0 или $\hat{0}$. Поэтому если $(f(u), f(v)) \in \{(1, 0),\, (\hat{1}, 0),\, (1, \hat{0}),\, (\hat{1}, \hat{0})\}$, то $c[t,\, f] = c[t',\, f|_{v \to \hat{0}}]$. Аналогично для симметричного случая.

Кроме того, ребро может влиять на наличие паросочетания среди вершин, покрашенных в 1. Пусть $(f(u), f(v)) = (1, 1)$. Тогда в оптимальном выборе $D \subset G_t$, либо в $D \setminus f^{-1}(\hat{1})$ существует паросочатание, не затрагивающее ребро uv, либо uv принадлежит паросочетанию. Тогда верно $c[t,\, f] = \min(c[t',\, f],\, c[t',\, f|_{u,v \to \hat{1}}])$. Итого:

$$c[t,\, f] = \begin{cases} c[t',\, f|_{v \to \hat{0}}], & (f(u), f(v)) \in \{(1, 0),\, (\hat{1}, 0),\, (1, \hat{0}),\, (\hat{1}, \hat{0})\} \\ c[t',\, f|_{u \to \hat{0}}], & (f(u), f(v)) \in \{(0, 1),\, (\hat{0}, 1),\, (0, \hat{1}),\, (\hat{0}, \hat{1})\} \\ \min(c[t',\, f],\, c[t',\, f|_{u,v \to \hat{1}}]), & (f(u), f(v)) = (1, 1) \\ c[t',\, f], & \text{иначе} \end{cases}$$

Forget node

Вершина имеет вид $X_t = X_{t'} \setminus \{v\}$. Оптимальное $D \subset G_t$ либо содержит v, либо нет. В первом случае v автоматически оказывается продоминированной, во втором случае, она должна быть продоминирована другой вершиной из D, поэтому верно следующее:

$$c[t,\, f] = \min(c[t',\, f|_{v\to 1}],\, c[t',\, f|_{v\to 0}])$$

3.2. Динамическое программирование для join node

Вершина имеет вид $X_t = X_{t_1} = X_{t_2}$, $G_t = G_{t_1} \cup G_{t_2}$ для детей t_1, t_2. Рассмотрим f_1, f_2 - покраски X_t. Назовём их согласованными с f, если

- $f(x) = \hat{1} \Rightarrow f_1(x) = f_2(x) = \hat{1}$
- $f(x) = \hat{0} \Rightarrow f_1(x) = f_2(x) = \hat{0}$
- $f(x) = 0 \Rightarrow \left[\begin{array}{l} f_1(x) = 0,\, f_2(x) = \hat{0} \\ f_1(x) = \hat{0},\, f_2(x) = 0 \end{array}\right.$
- $f(x) = 1 \Rightarrow \left[\begin{array}{l} f_1(x) = 1,\, f_2(x) = \hat{1} \\ f_1(x) = \hat{1},\, f_2(x) = 1 \end{array}\right.$

Легко видеть, что если $D \subset G_t$ - выполняет условия для f, то можно взять согласованные с f покраски f_1, f_2 такие, что $D \cap G_{t_1}$, $D \cap G_{t_2}$ будут выполнять условия для f_1, f_2 соответственно. А именно, если $f(v) = 0$, то вершина продоминирована какой-то вершиной $w \in D$. Если $w \in G_{t_1}$, положим $f_1(v) = 0$, $f_2(v) = \hat{0}$, иначе положим $f_1(v) = \hat{0}$, $f_2(v) = 0$. Если же $f(v) = 1$, то она имеет пару $w \in D \setminus f^{-1}(\hat{1})$. Если $w \in G_{t_1}$, положим $f_1(v) = 1$, $f_2(v) = \hat{1}$, иначе положим $f_1(v) = \hat{1}$, $f_2(v) = 1$.

Обратно, имея согласованные функции f, f_1, f_2, а также $D_1 \subset G_{t_1}$, $D_1 \subset G_{t_1}$, выполняющие условия для f_1, f_2 соответственно, множество $D = D_1 \cup D_2$ будет выполнять условия для f. Таким образом, можно написать следующее соотношение:

$$c[t, f] = \min\{c[t_1, f_1] + c[t_2, f_2] - |f^{-1}(1, \hat{1})|\},$$

где минимум берется по всем парам $f_1,\ f_2$, согласованных с f.

Для того, чтобы вычислить функцию за $\mathcal{O}^*\left(4^{|X_t|}\right)$, понадобится ввести следующее понятие:

Определение 3.2. Пусть задано множество X, и функции $f, g:\ 2^X \to \mathbb{N}$. Их сверткой называется функция $(f * g)(Y) = \min\limits_{Z \subset Y}(f(Z) + g(Y \setminus Z))$ для всех $Y \subset X$.

Теорема 15 (Глава 10 из [22])**.** Имея все $2^{|X|}$ значений для функций f, g, можно вычислить $f * g$ для всех подмножеств за время $\mathcal{O}^*\left(2^{|X|}\right)$.

Лемма 16. Зная все значения функционала c на вершинах $t_1,\ t_2$, можно вычислить все значения c на t за $\mathcal{O}^*\left(4^{|X_t|}\right)$.

Доказательство. Зафиксируем разбиение $X_t = f^{-1}\{0, \hat{0}\} \cup f^{-1}\{1, \hat{1}\} = X_0 \cup X_1$. Нам достаточно вычислить $c[t,\ f]$ для всех покрасок с данным разбиением за $\mathcal{O}^*\left(2^{|X_t|}\right)$, поскольку самих таких разбиений $2^{|X_t|}$. Заметим, что для согласованных покрасок f, f_1, f_2 разбиение на $X_0,\ X_1$ будет совпадать. Чтобы воспользоваться сверткой, с каждой покраской надо связать подмножество X_t. Сделаем это следующим образом: сопоставим покраске $S(f) = f^{-1}\{0,\ 1\} \subset X_t$. Легко видеть, что $S(f) = S(f_1) \cup S(f_2)$, $S(f_1) \cap S(f_2) = \emptyset$ для согласованных покрасок. Кроме того, поскольку $X_0,\ X_1$ фиксировано, любая покраска однозначно определяется по S: $f^{-1}(0) = S \cap X_0,\ f^{-1}(\hat{0}) = X_0 \setminus S$, и аналогично для $1,\ \hat{1}$. Обозначим такую покраску $f(S)$. Тогда верна следующая формула:

$$c[t, f(S)] = \min_{S' \subset S}\{c[t_1, f(S')] + c[t_2, f(S \setminus S')]\} - |X_1|$$

Тогда для функций $c(S) = c[t,\ f(S)], c_1(S) = c[t_1,\ f(S)], c_2(S) = c[t_2,\ f(S)]$, имеем свёртку $c = c_1 * c_2 - |X_1|$, которую можно вычислить за $\mathcal{O}\left(2^{|X_t|}\right)$. Исходная функция выражается как $c[t,\ f] = c(S(f))$. □

Теорема 17. Пусть дан граф G и его древесная декомпозиция ширины k. Тогда задача PDS может быть решена за время $\mathcal{O}^*\left(4^k\right)$, или, более точно, $\mathcal{O}\left(4^k \cdot k^{\mathcal{O}(1)} \cdot n\right)$.

Доказательство. Мы показали, что для каждой вершины декомпозиции можно вычислить c за время $\mathcal{O}^*\left(4^k\right)$. Так как всего вершин в декомпозиции $\mathcal{O}(kn)$, алгоритм будет работать за $\mathcal{O}\left(4^k \cdot k^{\mathcal{O}(1)} \cdot n\right)$. Для того, чтобы получить ответ на задачу, нужно взять $\min(c[root, f])$ по всем покраскам, красящим только в 0 или 1. □

3.3. Выводы

Был получен алгоритм, параметризованный древесной шириной, имеющий время работы $\mathcal{O}^*\left(4^k\right)$.

4. Оптимальность алгоритма, параметризованного древесной шириной

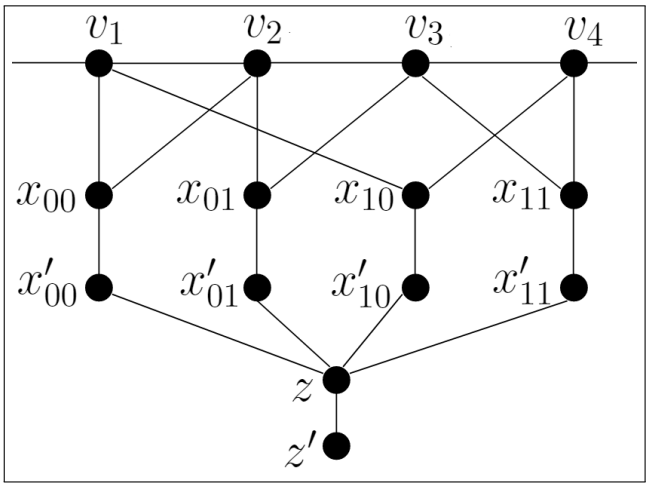
В данной секции будет показана оптимальность времени работы $\mathcal{O}^*\left(4^k\right)$ алгоритма, параметризованного шириной древесной декомпозиции в предположении истинности гипотезы SETH. Напомним, что из SETH слеует, что для задачи SAT не существует алгоритма, работающего за время $\mathcal{O}^*\left((2-\varepsilon)^n\right)$, где n - количество переменных в формуле. Для того, чтобы воспользоваться этим, по заданной формуле F из n переменных и m клозов мы построим (за полиномиальное время) граф G вместе с древесным разложением ширины $k = \frac{n}{2} + C$, такой, что наличие в нём PDS определённого размера будет равносильно наличию у F выполняющего набора. Тогда, имея алгоритм, решающий PDS за время $\mathcal{O}^*\left(4^{\lambda k}\right)$, где $\lambda < 1$, мы бы получили алгоритм для SAT, работающий за время $\mathcal{O}^*\left(4^{\lambda(n/2+C)}\right) = \mathcal{O}^*\left(2^{\lambda n}\right) = \mathcal{O}^*\left((2-\varepsilon)^n\right)$.

4.1. Построение графа по булевой формуле

Разобьём переменные формулы на пары. Каждой паре будет соответствовать длинная цепочка вершин. Каждая цепочка будет разбита на группы по 4 вершины. Рассмотрим такую группу, состоящую из вершин v_1, v_2, v_3, v_4.

Построение гаджета

Рассмотрим все 4 означивания пары переменных - 00, 01, 10, 11, и сопоставим каждому пару вершин (v_3, v_4), (v_1, v_4), (v_2, v_3), (v_1, v_2) соответственно. Добавим 10 вершин как изображено на рисунке. Каждму означиванию соответствуют вершины x и x'. При этом x связана ребром только с верши-



нами v, не соответствующими данному означиванию. Рассмотрим один произвольный клоз C, и заведём для него отдельную вершину C. Одной такой вершине будет соответствовать $\frac{n}{2}$ гаджетов - по одному для каждой пары переменных, назовём это группой гаджетов, соответствующей C. В каждом гаджете соединим вершины x' с C в том случае, если соответствующее означивание пары переменных выполняет клоз C.

Построение графа

Все $\frac{n}{2}$ четвёрок для i-го клоза соединим последовательно с соответствующими четверками для $(i+1)$-го клоза $(i < m)$. Получим $\frac{n}{2}$ цепочек длины $4m$, и $\frac{n}{2} \cdot m$ гаджетов. Далее всю конструкцию, включая вершины C_i скопируем $\frac{3}{2}n + 1$ раз, и также соединим последовательно. Добавим вершины l, l', r, r', и рёбра (l, r), (l, l'), (r, r'). Далее, крайние слои вершин (отмечено голубым на рисунке 1) обьединим в клики, и соединим соответственно с l и r.

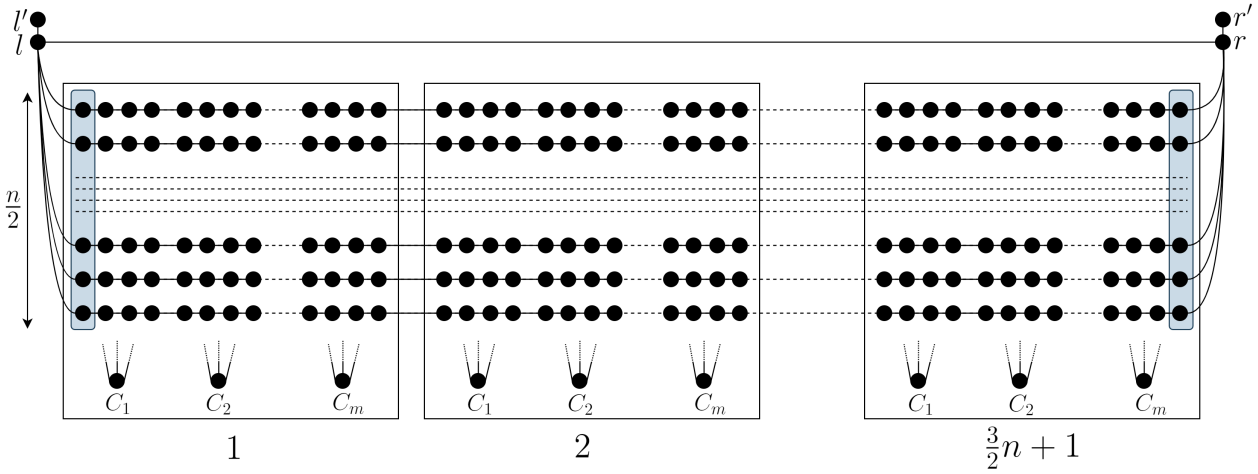


Рис. 1: Общая схема построенного графа

На этом построение графа заканчивается, и можно сформулировать следующее утверждение:

Лемма 18. У F есть выпоняющее означивание тогда и только тогда, когда в построенном графе G существует PDS размера $2nm\left(\frac{3}{2}n + 1\right) + 2$.

4.2. Выполнимость влечёт наличие PDS

Пусть у F есть выполняющее означивание. Требуется получить парное доминирующее множество D размера $2nm\left(\frac{3}{2}n+1\right)+2$. Во первых, возьмём в D вершины l, r. Для каждой пары переменных возьмём в каждой цепочке соответствующие вершины в множество D. Например, пусть переменные означены в 00. Тогда в каждой группе v_3, v_4 будут принадлежать нашему множеству. Рассмотрим произвольный гаджет из этой цепочки. Из вершин x не продоминированной осталась только x_{00}. Возьмём в D вершины x'_{00} и z. Легко видеть, что при любом означивании все вершины (кроме, возможно, v_1, v_4) гаджета оказались продоминированы. В случае, когда какая-то вершина, например v_1 не продоминирована другой вершиной из гаджета, она обязательно продоминирована вершиной v_4 из соседнего гаджета. Для крайних гаджетов, не имеющих соседнего гаджета, вершины будут продоминированы l или r. Кроме того, поскольку означивание выполняет каждый клоз, то для любой вершины C она будет продоминирована какой-то вершиной x', соответствующей означиванию некоторой пары переменных, которое выполняет клоз C.

Осталось показать, почему в D будет совершенное паросочетание. Поскольку для фиксированной цепочки в каждой группе выбраны одни и те же вершины, они будут разбиты на пары подряд идущих. Исключение могут представлять крайние вершины, не имеющие пары в отсутствующей соседней группе. Это возможно только если в цепочке выбраны вершины v_1, v_4. Пусть без пары остаются некоторые вершины v_1, соседние вершине l. В таком случае, можно разбить их на пары между собой, поскольку все такие вершины мы соединили в клику. Если их чётное число, то остаётся сказать, что l и r соединены ребром. Если их нечётное число, то единственную вершину без пары спарим с l, а симметричную вершину из соседей r спарим с r.

В каждом гаджете выбрано 4 вершины, всего гаджетов $\frac{n}{2}\cdot m\cdot\left(\frac{3}{2}n+1\right)$. Плюс вершины l, r, итого получается $|D|=2nm\left(\frac{3}{2}n+1\right)+2$.

4.3. Наличие PDS влечёт выполнимость

Пусть в графе есть парное доминирующее множество размера $|D|= 2nm\left(\frac{3}{2}n+1\right)+2$. Во-первых, оно содержит l, r, так как l', r' должны быть продоминированы. Несложно убедиться, что в каждом гаджете хотя-бы 4 вершины должны принадлежать D. Тогда, из размера D следует, что в каждом гаджете выбрано ровно 4 вершины, которые, ко всему прочему, имеют определённый вид: $z \in D$, поскольку z' должна быть проминирована, z имеет соседа x' для какого-то означивания (например x'_{00}). Непродоминированными остаются x_{01}, x_{10}, x_{11}. Если какая-то из них принадлежит D (например x_{01}), то единственный способ продоминировать x_{10}, x_{11} это взять вершину v, соседнюю им обеим, то есть v_4. Заметим, что каждая вершина v имеет только двух соседей среди вершин x, поэтому x_{01} и v_4 не образуют пару. Значит как минимум 2 вершины v принадлежат D. Только пара вершин v_3, v_4, не смежных с x_{00} способна продоминировать все три оставшиеся вершины x.

Получается, что каждый гаджет задаёт означивание соответствующей пары переменных. А вершины D во всей группе гаджетов, соответствующей клозу C задают общее означивание, выполняющее клоз C, поскольку вершина C продоминирована. Однако нет гарантии, что группы, соответствующие разным клозам задают одно и то же означивание. Однако верно следующее: найдутся m подряд идущих групп, в которых для каждой фиксированной цепочки в разных группах будут выбраны одни и те же вершины. Тогда будет справедливо сказать, что они задают одно и то же означивание, которое выполняет все m клозов.

Для того, чтобы понять это, рассмотрим фиксированную цепочку. В ней идут соседние пары вершин из D. Поскольку все вершины v продоминированы, расстояние между двумя соседними парами не более двух вершин. Если в первой группе взята пара v_1, v_4, то в каждой оставшейся группе также обязана быть взята пара v_1, v_4. Иначе в каждой группе взято ребро, начинающееся с v_1, v_2 или v_3. Если в какой-то группе взято ребро, начинающееся в v_i, то в следующей группе взятое ребро должно начинаться с v_j, $j \leq i$, так как иначе между ними было расстояние бы

в три вершины. Количество плохих групп, то есть в которых j строго меньше i, не более трёх. Суммарно во всех цепочках таких групп не более $\frac{3}{2}n$. Всего имеется $\frac{3}{2}n + 1$ участок длины m, поэтому найдется участок из m групп, в котором не будет плохих групп. Этот участок задаст требуемое означивание.

4.4. Древесная декомпозиция полученного графа

Несложно проверить, что построенный граф имеет древесную ширину $\frac{n}{2}+C$. Для этого явно предьявим разложение ширины $\frac{n}{2}+C$, которое будет представлять из себя путь длины $\mathcal{O}(mn^2)$, вершинам которого будут соответствовать множества X_i. Во-первых, в каждом X_i будут содержаться вершины l, l', r, r'. Кроме того, в каждом X_i будет содержаться по одной вершине v_1 или v_4 из каждой цепочки, кроме, возможно одной, а также не более одного целого гаджета и не более одной вершины C_i. Таким образом размер $|X_i|$ будет не больше, чем $4 + \frac{n}{2} + 14 + 1 = \frac{n}{2} + 19$.

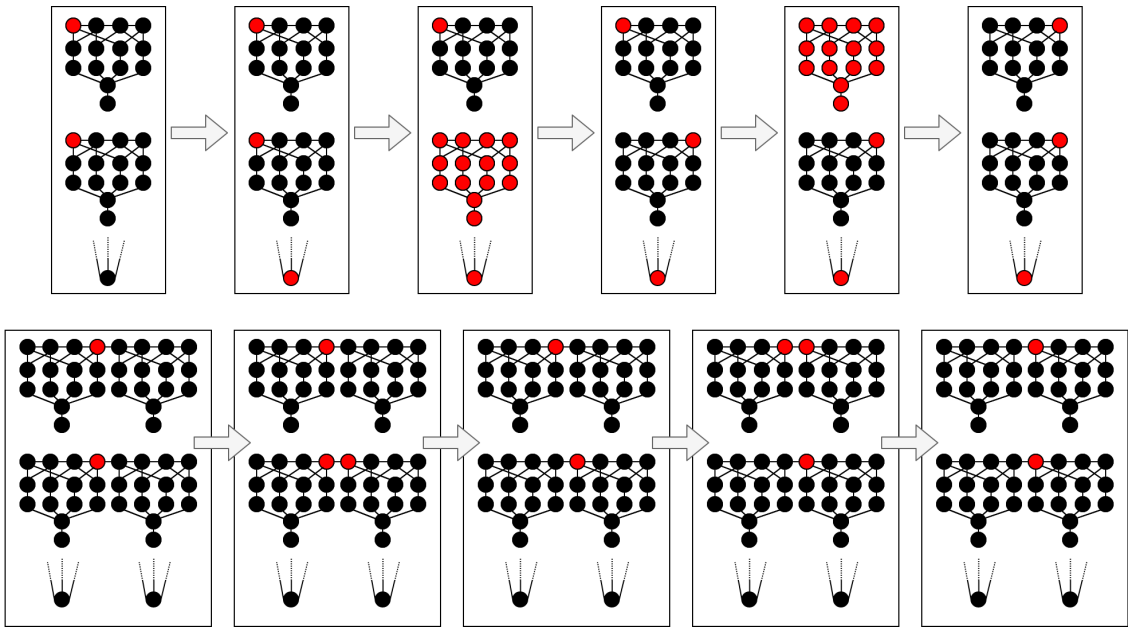


Рис. 2: Построение древесной декоспозиции

Для начала возьмём в X_0 все начальные вершины v_1 (соседей l) и вершины l, l', r, r'. Далее добавим вершину C_1, и гаджет соответствующий первой цепочке. Затем оставим в первой цепочке только вершину

v_4. После этого сделаем последовательно тоже самое с оставшимися цепочками, получив X_i состоящее из l, l', r, r', C_1 и вершин v_4 из первой группы, соответствующей C_1. После этого выбросим C_1, а затем для каждой цепочки последовательно добавим следующую вершину v_1 для имеющейся, и выбросим имеющуюся v_4. Дальше добавим C_2 и продолжая также, как и изначально, дойдём до конца графа.

Подводя итог, можно сформулировать следующую теорему:

Теорема 19. Предположим, что гипотеза SETH верна. Пусть дан граф G, и его древесная декомпозиция ширины k. Для любого $\varepsilon > 0$ не существует алгоритма, решающего задачу PDS за время $\mathcal{O}^*\left((4-\varepsilon)^k\right)$.

4.5. Выводы

В предположении истинности гипотезы SETH доказана оптимальность алгоритма, параметризованного шириной k древесной декомпозиции, работающего за время $\mathcal{O}^*\left(4^k\right)$.

5. Параметризация расстоянием

В данном разделе будут описаны алгоритмы для задач DS, TDS и PDS, параметризованные расстоянием до различных классов графов. Пусть есть граф G, вершины которого представлены в обьединение множеств $V(G) = S \cup F$. Граф индуцированный F лежит в каком-то классе графов $\mathfrak{C}$, для которого существует полиномиальный алгоритм, решающий задачу DS, или её вариацию. S представляет из себя произвольный граф и называется модулятором. Размер $|S|$ выступает в качестве параметра и называется расстоянием до класса $\mathfrak{C}$.

5.1. Общая схема алгоритмов

Все представленные алгоритмы будут работать по следующей схеме: пусть требуется найти множество D, сделаем предположение о его пересечении с модулятором $D_1 = D \cap S$. После этого рассмотрим вершины S, оставшиеся непродоминированными: $S_1 = S \setminus N_S[D_1]$. При этом некоторые вершины F оказались продоминированными множеством D_1. Для решения DS остается выбрать $D \cap F = D_2$ так, чтобы S_1 и F было продоминировано. Можно сформулировать задачи как вариации задачи Set-Cover, рассмотрев вершины F как покрывающие множества, а S_1 как элементы универсума. Более точно определим следующие задачи:

Определение 5.1 ($SC(\mathfrak{C})$, $SC_t(\mathfrak{C})$, $SC_p(\mathfrak{C})$)**.** Дан граф G, и его разбиение на $F \in \mathfrak{C}$, D_1 и S_1. Причём между D_1 и S_1 нет рёбер.

- В задаче $SC(\mathfrak{C})$ требуется найти такое минимальное $D_2 \subset F$, что вершины из $(F \setminus N[D_1]) \cup S_1$ продоминированы D_2.
- В задаче $SC_t(\mathfrak{C})$ требуется найти такое минимальное $D_2 \subset F$, что вершины из $(F \setminus N[D_1]) \cup S_1$ продоминированы D_2, и кроме того в графе, индуцированном $D_1 \cup D_2$ нет изолированных вершин.
- В задаче $SC_p(\mathfrak{C})$ требуется найти такое минимальное $D_2 \subset F$, что вершины из $(F \setminus N[D_1]) \cup S_1$ продоминированы D_2, и кроме того в графе, индуцированном $D_1 \cup D_2$ есть совершенное паросочетание.

Для некоторых классов $\mathfrak{C}$ задачу $SC(\mathfrak{C})$ можно решить динамическим программированием по подмножествам S_1 за $\mathcal{O}^*\left(2^{|S_1|}\right)$. Например в [10] представлен подобный алгоритм для случая $\mathfrak{C}$ - кластерных графов.

Теорема 20. Если $SC(\mathfrak{C})$ можно решить за время $\mathcal{O}^*\left(2^{|S_1|}\right)$, то DS можно решить за $\mathcal{O}^*\left(3^{|S|}\right)$.

Доказательство. Алгоритм перебирает все подмножества D_1. Для каждого из них берёт $S_1 = (S \setminus N_S[D_1]) \subset (S \setminus D_1)$, и решает $SC(\mathfrak{C})$. Итоговое время работы $\sum\limits_{D_1 \subset S} \mathcal{O}^*\left(2^{|S_1|}\right) = \mathcal{O}^*\left(3^{|S|}\right)$. □

В случае же $SC_t(\mathfrak{C})$, $SC_p(\mathfrak{C})$ приходится рассматривать динамику по подмножествам всего S для каждого фиксированного D_1, что увеличивает время работы с $3^{|S|}$ до $4^{|S|}$:

Теорема 21. Если $SC_t(\mathfrak{C})$ / $SC_p(\mathfrak{C})$ можно решить за время $\mathcal{O}^*\left(2^{|S_1 \cup D_1|}\right)$, то TDS / PDS можно решить за $\mathcal{O}^*\left(4^{|S|}\right)$.

Доказательство. Алгоритм перебирает все подмножества D_1. Для каждого из них берёт $S_1 = (S \setminus N_S[D_1]) \subset (S \setminus D_1)$, и решает $SC_t(\mathfrak{C})$ или $SC_p(\mathfrak{C})$ соответственно. Итоговое время работы $\sum\limits_{D_1 \subset S} \mathcal{O}^*\left(2^{|S_1 \cup D_1|}\right) = 2^{|S|} \times \mathcal{O}^*\left(2^{|S|}\right) = \mathcal{O}^*\left(4^{|S|}\right)$. □

Далее будут представлены описанные алгоритмы для кластерных графов в случае PDS, а также для интервальных графов и графов перестановок для случая DS и TDS.

5.2. Решение SC_p(cluster graphs)

Кластерные графы представляют из себя несвязное обьединение клик. Равносильное определение - отсутствие в них индуцированных путей длины 2. Рассмотрим граф F, состоящий из клик. Некоторые клики могли быть продоминированы вершинами из D_1. Для того, чтобы оставшиеся клики были продоминированы, в каждой из них достаточно выбрать хотя-бы одну вершину. Упорядочим вершины в любом порядке, в

котором клики идут последовательно. Обозначим за $C(i)$ клику, которой принадлежит вершина i. Рассмотрим функцию для $i = 0, 1, ..., |F|$, $x, y \in \{0, 1\}$, $S' \subset S_1$, $D' \subset D_1$:

Определение 5.2. $PDS(i, x, y, S', D')$ - минимальное подмножество $D \subset \{1, ..., i\}$, доминирующее вершины из S', такое, что:

- для каждой клики $C < C(i)$, которая не полностью доминируется D_1, в ней есть хотя-бы одна вершина, принадлежащая D;
- если $x = 1$, то D принадлежит хотя-бы одна вершина из $C(i)$;
- если $y = 0$, то в $D \cup D'$ есть совершенное паросочетание;
- если $y = 1$, то в $D \cup D'$ есть паросочетание, покрывающее все вершины, кроме ровно одной вершины, принадлежащей $C(i)$.

Если такого не существует, обозначим значение за N.

Легко видеть, что ответом на задачу является $PDS(|F|, x, 0, S_1, D_1)$, где $x = 1$ если последняя клика не полностью продоминирована D_1. Пусть требуется вычислить $D = PDS(i, x, y, S', D')$.

Случай 1

Вершина i - не первая вершина в своей клике. Тогда i может как принадлежать, так и не принадлежать D. Если она не принадлежит D, то $D = PDS(i-1, x, y, S', D')$. Если она принадлежит D, то возможны случаи:

1. i имеет пару j среди своих соседей в D', и тогда
   $D = \{i\} \cup PDS(i-1, 0, y, S' \setminus N_{S_1}(i), D' \setminus \{j\})$.
2. i имеет пару $j < i$ внутри $C(i)$. Если $y = 0$, то в $(D \cap C(i)) \setminus \{i\}$ должна остаться одна неспаренная вершина j. Если $y = 1$, то в $D \cap C(i)$ есть одна неспаренная вершина k. Тогда в $(D \cap C(i)) \setminus \{i\}$ должны остаться две неспаренные вершины j и k. В таком случае можно считать их спаренными, так как они лежат в одной клике, а вершину i - неспаренной, и тогда
   $D = \{i\} \cup PDS(i-1, 0, 1-y, S' \setminus N_{S_1}(i), D')$.
3. $y = 1$ и i - единственная неспаренная вершина в клике. Случай анологичен предыдущему.

Итого,

$$PDS(i,\, x,\, y,\, S',\, D') = \min\{PDS(i-1,\, x,\, y,\, S',\, D'),$$
$$\{i\} \cup PDS(i-1,\, 0,\, y,\, S' \setminus N_{S_1}(i),\, D' \setminus \{j\}) \text{ for } j \in N_{D'}(i),$$
$$\{i\} \cup PDS(i-1,\, 0,\, 1-y,\, S' \setminus N_{S_1}(i),\, D')\}$$

Случай 2

Вершина i - первая вершина в своей клике. Тогда $C(i-1) \neq C(i)$. Обозначим $x_1 = 1$ если $i > 1$ и $C(i-1)$ не полностью продоминирована D_1, и $x_1 = 0$ иначе.

Пусть $x = 0$ и $y = 0$. Тогда мы можем не взять i в D. В таком случае $D = PDS(i-1,\, x_1,\, 0,\, S',\, D')$. Если мы её берём, альтернатива аналогична случаю $y = 0$, $x = 1$: i спарена с одним из соседей j в D'. Аналогично случаю 1, $D = \{i\} \cup PDS(i-1,\, x_1,\, 0,\, S' \setminus N_{S_1}(i),\, D' \setminus \{j\})$.

Если $y = 1$, то i принадлежит D и является единственной неспаренной вершиной там. Тогда $D = \{i\} \cup PDS(i-1,\, x_1,\, 0,\, S' \setminus N_{S_1}(i),\, D')$.

База рекурсии при $i = 0$: $PDS(0, x, y, S', D') = \emptyset$ если и только если $S' = \emptyset$, а в D' есть совершенное паросочетание, иначе N.

5.3. Интервальные графы

Граф G называется интервальным, если его вершины можно соотнести с семейством отрезков вещественной прямой так, что ребро соответствует пересечению отрезков. В 1976 в [2] был представлен алгоритм для распознавания интервальных графов, и построения явного соответствия, рабтающий за $\mathcal{O}^*(|V(G)|+|E(G)|)$. Существуют полиномиальные алгоритмы, решающие DS для интервальных графов ([3]). Следующие результаты являются адаптацией алгоритмов для DS и TDS, представленных в [14].

5.3.1. Необходимые определения

Напомним, что в задачах SC, SC_t, SC_p даны F - интервальный граф, D_1, доминирующее часть вершин F, а также S_1, которое должно быть

продоминировано вершинами из F. Обозначим $F' = N_F(D_1)$ - вершины, которые не обязаны быть продоминированы. Для F зафиксируем соответствие вершин и отрезков, а также порядок вершин в соответствии с увеличением правой границы. Пронумеруем их, начиная с единицы. Введём несколько понятий:

Определение 5.3. $L(i) = \{j \leq i : (i,j) \in E\}$ - все отрезки, правая граница которых лежит в i.

Определение 5.4. $l(i) = \min(L(i)) = \min\{j \leq i : (i,j) \in E\}$ - отрезок с самой левой правой границей, лежащей в отрезке i. Заметим, что $L(i) = \{l(i), ..., i\}$.

Определение 5.5. Пусть $i \notin F'$. Рассмотрим вершины j из $L(i) \setminus F'$, и выберем вершину $k(i)$ с максимальным $l(j) : k(i) = \text{argmax}\{l(j) : j \in L(i) \setminus F'\}$. Заметим, что $l(k(i)) \geq l(i)$, поскольку $i \in L(i) \setminus F'$.

Лемма 22. Пусть $k = k(i)$. Каждая вершина $a \in [l(k), ..., i]$ доминирует все вершины $b \in [l(a), ..., i] \setminus F'$.

Доказательство. Ясно, что a доминирует все вершины из $[l(a), ..., a]$. Пусть есть $l(k) \leq a < b \leq i$, такие что $(a,b) \notin E$, $b \notin F'$. Это значит, что левая граница b лежит правее правой границы a, и $l(k)$, из чего следует $l(b) > l(k)$, что противоречит определению $k(i)$. □

5.3.2. Решение SC(interval graphs)

В задаче SC нам даны F, D_1 и S_1 (далее обозначено за S), требуется найти минимальное $D \subset F$, доминирующее S и $F \setminus F'$. Пусть $S' \subset S$. Определим функцию:

Определение 5.6. $DS(i,\, S')$ - минимальное подмножество F, доминирующее вершины из S', а также вершины $j \leq i$, не входящие в F'. Отметим, что оно может содержать вершины $j > i$.

Ясно, что ответом на задачу будет являться $DS(|F|,\, S)$. Поймём, как рекурсивно выразить величину $DS(i,\, S')$. Пусть $i \in F'$. Тогда ясно, что

$DS(i,\, S') = DS(i-1,\, S')$. Поэтому можно считать, что $i \notin F'$. Пусть $k = k(i)$. Поскольку $k \notin F'$ должно быть продоминировано, один из соседей $j \in N_F[k]$ принадлежит $DS(i,\, S')$. Если вершина $j \leq i$, по лемме 22 она доминирует $[l(j), ..., i] \setminus F'$. Если $j > i$, то она тоже доминирует $[l(j), ..., i]$, так как покрывает все правые границы этих отрезков. Тогда

$$DS(i,\, S') = \min\{\{j\} \cup DS(l(j)-1,\, S' \setminus N_S(j)) :\, j \in N_F[k]\}.$$

Остаётся понять, чему равно $DS(0,\, S')$. Это минимальное множество $D \subset F$, которое доминирует S' - задача Set Cover, которую для всех $S' \subset S$ можно решить с помощью динамического программирования за $\mathcal{O}^*\left(2^{|S|}\right)$. Итого, имеем $|F| \times 2^{|S|}$ состояний, каждое из которых пересчитывается через предыдущие за полиномиальное время. Итоговое время работы - $\mathcal{O}^*\left(2^{|S|}\right)$.

5.3.3. Решение SC_t(interval graphs)

Будем говорить, что множество A *тотально* доминирует множество B, если $\forall v \in B\, \exists w \in A :\, (v, w) \in E(G)$. В задаче SC_t даны F, D_1 и S_1. Среди вершин D_1 возьмём изолированные вершины $D_1' \subset D_1$. Заметим, что $D_1 \setminus D_1'$ уже продоминировано самим собой. Требуется найти $D_2 \subset F$, которое доминирует D_1' и S_1, а также тотально доминирует $F \setminus F'$. Обозначим $S = S_1 \cup D_1'$. Теперь требуется найти минимальное $D \subset F$, доминирующее S и тотально доминирующее $F \setminus F'$. Определим следующие функции:

Определение 5.7. $TDS(i,\, S')$ - минимальное подмножество F, доминирующее вершины из S', а также тотально доминирующее вершины $j \leq i$, не входящие в F'.

Определение 5.8. $PartialTDS(i,\, S')$ - минимальное подмножество F, доминирующее вершины из S', а также тотально доминирующее вершины $(\{i\} \cup \{j \leq l(i) - 1\}) \setminus F'$.

Поймём, как рекуррентно выразить $PartialTDS(i,\, S')$. Так как i должна быть продоминирована, один из соседей $j \in N_F(i)$ обязан быть

взят. Непродоминированными остаются вершины $l \leq \min(l(i),\, l(j)) - 1$. Поэтому

$$PartialTDS(i,\, S') = \min\{\{j\} \cup TDS(\min(l(i),\, l(j))-1,\, S' \setminus N_S(j)) : \\ j \in N_F(i)\}.$$

Для получения формулы для $TDS(i,\, S')$, как и раньше, можно считать, что $i \notin F'$, иначе $TDS(i,\, S') = TDS(i-1,\, S')$. Возьмём $k = k(i)$. Один из соседей $j \in N_F(k)$ принадлежит $TDS(i,\, S')$. Как и раньше, если $j \leq i$, она доминирует все вершины из $[l(j), ..., i] \setminus F'$, кроме самой себя. Тогда $TDS(i,\, S') = \{j\} \cup PartialTDS(j,\, S' \setminus N_S(j))$. Если же $j > i$, вершина j не обязана быть продоминирована, и можно заключить, что $TDS(i,\, S') = \{j\} \cup TDS(l(j)-1,\, S' \setminus N_S(j))$.

$$TDS(i,\, S') = \min(\{\{j\} \cup PartialTDS(j,\, S' \setminus N_S(j)) :\, j \in N_F[k] \cap L(i)\} \cup \\ \cup \{\{j\} \cup TDS(l(j)-1,\, S' \setminus N_S(j)) :\, j \in N_F[k] \setminus L(i)\}).$$

Отметим, что $PartialTDS(0,\, S')$ не определено и не используется в алгоритме. Определение $TDS(0,\, S')$ не накладывает условий на доминирование F, и, как и в случае DS, совпадает с задачей Set Cover. Итоговое время работы $2^{|S|}$, где $S = S_1 \cup D'_1$.

5.4. Графы перестановок

Граф G называется графом перестановки π, если пара вершин i, j соединены ребром если и только если $(i-j)(\pi(i) - \pi(j)) < 0$, то есть если π меняет местами i и j. Удобно себе представлять граф в виде двух колонок из $|V|$ точек, в которой проведены отрезки из i в $\pi(i)$. Вершины графа будут соответствовать точкам левой колонки. Тогда две вершины соединены ребром в G если соответствующие отрезки пересекаются. В 1983 [26] был представлен алгоритм для распознавания графа перестановки и построения самой перестановки π, работающий за $\mathcal{O}(|V(G)|^2)$. Далее будет представлен алгоритм, решающий $SC(\text{permutation graphs})$ за $\mathcal{O}(|F|^3 \times 2^{|S_1|})$, являющийся адаптацией алго-

ритма для DS, представленном в [9], а также аналогичный алгоритм для SC_t(permutation graphs).

5.4.1. Решение SC(permutation graphs)

Напомним, что можно считать, что в SC помимо F нам дано множество S вершин которые надо продоминировать, а также $F' \subset F$, которые не обязаны быть продоминированы.

Определение 5.9. За $V(i,j)$ обозначим множество $\{1,...,i\} \cap \pi^{-1}\{1,...,j\}$. Визуально это отрезки, левая граница которых не больше i, а правая не больше j.

Определение 5.10. За $DS(i,j,k,S')$ для $i,j,k \in [0, |F'|]$ и $S' \subset S$ обозначим минимальное множество $D \subset \{1,...,i\}$, которое доминирует $V(i,j) \setminus F'$ и S', а также для которого выполнено $\max\{\pi(l) : l \in D\} \geq k$ (при пустом D максимум считается нулём). Если такого не существует, обозначим значение за N.

Ясно, что ответом на задачу будет являться $DS(|F|, |F|, 0, S)$. Пусть необходимо вычислить $D = DS(i,j,k,S')$. Рассмотрим возможные случаи:

1. $i \in D$. Вершина доминирует все вершины $l < i$, для которых $\pi(l) > \pi(i)$, и не доминирует те, для которых $\pi(l) < \pi(i)$.
   1.1. $\pi(i) \geq k$. Тогда $D = \{i\} \cup DS(i{-}1,\ \min(j, \pi(i){-}1),\ 0,\ S' \setminus N_S(i))$.
   1.2. $\pi(i) < k$. Тогда $D = \{i\} \cup DS(i{-}1,\ \min(j, \pi(i){-}1),\ k,\ S' \setminus N_S(i))$.
2. $i \notin D$.
   2.1. $i \in F'$ или $\pi(i) < j$. Тогда $D = DS(i{-}1,\ j,\ k,\ S')$, так как она не обязана быть продоминирована.
   2.2. $i \notin F'$ и $\pi(i) \geq j$. Вершина должна быть продоминирована, так что $D = DS(i{-}1,\ j,\ \max(k, \pi(i)),\ S')$.

Функция считается, как минимальное D из случаев 1 и 2. В каждом случае мы уменьшаем i. Очевидно, $DS(0,j,k,S') = N$ если $S' \neq \emptyset$ или $k > 0$, и $\emptyset$ иначе.

5.4.2. Решение SC_t(permutation graphs)

Как и в случае SC_t(interval graphs), можно считать, что нам даны F, $F' \subset F$ и S, и требуется найти минимальное $D \subset F$, доминирующее S и тотально доминирующее $F \setminus F'$. Отличия от предыдущего алгоритма будут минимальны.

Определение 5.11. За $TDS(i, j, k, S')$ для $i, j, k \in [0, |F'|]$ и $S' \subset S$ обозначим минимальное множество $D \subset \{1, ..., i\}$, которое **тотально** доминирует $V(i, j) \setminus F'$ и доминирует S', а также для которого выполнено $\max\{\pi(l) : l \in D\} \geq k$ (при пустом D максимум считается нулём). Если такого не существует, обозначим значение за N.

Как и ранее, ответом является $TDS(|F|, |F|, 0, S)$. Пусть необходимо вычислить $D = TDS(i, j, k, S')$. Рассмотрим возможные случаи:

1. $i \in D$.
   - 1.1. $\pi(i) \geq k \Rightarrow D = \{i\} \cup TDS(i{-}1, \min(j, \pi(i){-}1), \boldsymbol{\pi(i)}, S' \setminus N_S(i))$, поскольку $D \setminus \{i\}$ обязано содержать вершину l, доминирующую i, то есть у которой $\pi(l) > \pi(i)$.
   - 1.2. $\pi(i) < k$. Тогда $D = \{i\} \cup TDS(i{-}1, \min(j, \pi(i){-}1), k, S' \setminus N_S(i))$, сама i будет продоминировано вершиной $l \in D \setminus \{i\}$, у которой $\pi(l) > \pi(i)$.
2. $i \notin D$. Отличий от алгоритма для DS нет.
   - 2.1. $i \in F'$ или $\pi(i) < j$. Тогда $D = TDS(i{-}1, j, k, S')$.
   - 2.2. $i \notin F'$ и $\pi(i) \geq j$. Тогда $D = TDS(i{-}1, j, \max(k, \pi(i)), S')$.

База рекурсии аналогична алгоритму для SC(permutation graphs): $TDS(0, j, k, S') = N$ если $S' \neq \emptyset$ или $k > 0$, и $\emptyset$ иначе.

5.5. Потенциальное улучшение времени работы

Заметим, что в задачах вида $SC(\mathfrak{C})$ рассматриваются только пары подмножеств D_1 и $S_1 = S \setminus N_S[D_1]$, но не рассматривается $N_S[D_1] \setminus D_1$. В случае, когда $N_S[D_1]$ большое, это накладывает существенные ограничения на возможные случаи множеств: если в S много рёбер, то не так много пар D_1, S_1, не соединённых рёбрами. Наблюдение можно сформулировать в терминах максимального независимого множества S:

Теорема 23. Пусть дан граф G, представленный в виде обьединения S и $F \in \mathfrak{C}$. Пусть размер максимального независимого множества $i(S) < \lambda|S|$ для некоторого фиксированного $\lambda < 1$.

- Пусть $SC(\mathfrak{C})$ можно решить за время $\mathcal{O}^*\left(2^{|S_1|}\right)$. Тогда DS можно решить за время $\mathcal{O}^*\left((3-\varepsilon(\lambda))^{|S|}\right)$.
- Пусть $SC_t(\mathfrak{C})$ / $SC_p(\mathfrak{C})$ можно решить за время $\mathcal{O}^*\left(2^{|S_1 \cup D_1|}\right)$. Тогда TDS / PDS можно решить за $\mathcal{O}^*\left((4-\varepsilon(\lambda))^{|S|}\right)$.

Доказательство. Алгоритм для DS для каждого фиксированного D_1 работает динамикой по подмножествам $S_1' \subset S_1$, то есть фактически перебирает все пары D_1, $S_1' \subset S$, между которыми нет рёбер. Заметим, что имея произвольный алгоритм для решения $SC(\mathfrak{C})$ за время $\mathcal{O}^*\left(2^{|S_1|}\right)$, время работы асимптотически будет таким же:

$$\sum_{D_1 \subset S} 2^{|S_1|} = \sum_{D_1 \subset S} \sum_{S_1' \subset S_1} 1 = \sum_{D_1,\, S_1' \subset S} 1.$$

Поскольку между D_1 и S_1' нет рёбер, то если есть компонента связности $K \subset D_1 \cup S_1'$, она будет полностью сожержаться в D_1 или S_1'. Переберём сначала подмножества $X = D_1 \cup S_1'$. Для кажого из них будем перебирать разбиения на D_1, S_1'. Пусть для X в нём ровно $t(X)$ изолированных вершин. Компонент связности в X, в которых хотя бы 2 вершины, не более чем $\frac{|X|-t(X)}{2}$. Тогда время работы можно будет оценить как

$$\sum_{X \subset S} 2^{\left(\frac{|X|-t(X)}{2} + t(X)\right)} = \sum_{X \subset S} 2^{\frac{|X|+t(X)}{2}}. \tag{1}$$

В алгоритмах для TDS и PDS для фиксированного D_1 строится динамика по подмножествам $A \subset D_1 \cup S_1$. Тоесть рассматриваются тройки множеств D_1', $D_1'' \subset D_1$, $S_1' \subset S_1$, такие, что между S_1' и $D_1' \cup D_1''$ нет рёбер (здесь в качестве A выступает $D_1' \cup S_1'$). Рассуждения, аналогичные случаю DS, показывают, что произвольный алгоритм для $SC_t(\mathfrak{C})$ ($SC_p(\mathfrak{C})$) со временем $\mathcal{O}^*\left(2^{|S_1 \cup D_1|}\right)$ даст асимптотически такое же время, как динамика по подмножествам $D_1 \cup S_1$. Как и раньше, сначала можно перебирать $X = D_1' \cup D_1'' \cup S_1'$. Если есть компонента связности, содержащая $c_i \geq 2$ вершин, то возможно $2^{c_i} + 1 < 5^{\frac{c_i}{2}}$ вариантов распределения этих вершин по трём множествам: либо компонента полностью лежит в S_1', либо полностью в $D_1' \cup D_1''$. Время можно оценить как

$$\sum_{X \subset S} 5^{\left(\frac{|X|-t(X)}{2}\right)} 3^{t(X)} = \sum_{X \subset S} 3^{\mu|X|+(1-\mu)t(X)}, \tag{2}$$

где $\mu = \frac{\log_3(5)}{2} \approx 0.732$. Видно, что при $t(X) = |X|$ формулы вырождаются в $3^{|S|}$ и $4^{|S|}$ соответственно. Доказательство сводится к следующему утверждению, которое будет доказано далее: □

Лемма 24. Пусть в графе S размер максимального независимого множества $i(S) < \lambda|S|$. Тогда выражение 1 не превосходит $\mathcal{O}^*\left((3 - \varepsilon(\lambda))^{|S|}\right)$, а выражение 2 не превосходит $\mathcal{O}^*\left((4 - \varepsilon(\lambda))^{|S|}\right)$ соответственно.

Чтобы доказать для обоих случаев, напишем выражение в общем виде:

$$\sum_{X \subset S} M^{\mu|X|+(1-\mu)t(X)} \tag{3}$$

Здесь для первого случая $M = 2$, $\mu = \frac{1}{2}$, а для второго случая $M = 3$, $\mu = \frac{\log_3(5)}{2}$. Докажем, что выражение 3 не превосходит $\mathcal{O}^*\left((M + 1 - \varepsilon(\lambda))^{|S|}\right)$. Для этого вспомним некоторые факты из математического анализа:

Определение 5.12. Для числа $0 \leq x \leq 1$ энтропией $H(x)$ называется величина $-x\log_2(x) - (1-x)\log_2(1-x)$.

Лемма 25. Биномиальный коэффицент $\binom{n}{k}$ равен $2^{H(\frac{k}{n})}$ с точностью до полиномиального множителя.

Доказательство. $\binom{n}{k} = \frac{n!}{k!(n-k)!}$. По формуле стирлинга $n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$. □

Лемма 26. $\frac{d}{dx}H(x) = -\log_2(x) + \log_2(1-x) = \log_2\left(\frac{1-x}{x}\right)$.

Далее будем обозначать $|S| = n$. Запишем выражение 3 в следующем виде:

$$\sum_{b \in \{0, \frac{1}{n}, \frac{2}{n}, \ldots, 1\}} \sum_{X \subset S: |X| = b \cdot n} M^{\mu|X| + (1-\mu)t(X)} \tag{4}$$

Поскольку мы доказываем оценку с точностью до полиномиального множителя, достаточно доказывать утверждение только для внутренней суммы, так как внешняя сумма содержит всего n слагаемых.

Лемма 27. При $b \notin [\frac{M}{M+1} - \Delta(\varepsilon),\ \frac{M}{M+1} + \Delta(\varepsilon)]$ выполнено

$$\sum_{X \subset S: |X| = b \cdot n} M^{\mu|X| + (1-\mu)t(X)} < \binom{n}{bn} M^{bn} = \mathcal{O}^*\left((M + 1 - \varepsilon)^n\right).$$

Доказательство. Первое неравенство очевидно в силу $t(X) \leq |X|$. Из леммы 25 следует, что достаточно доказать

$$H(b) + \log_2(M)b < \log_2(M + 1 - \varepsilon). \tag{5}$$

Производная левой части по b равна $\log_2\left(\frac{1-b}{b}\right) + \log_2(M)$. Приравнивая к нулю, получаем $b_{max} = \frac{M}{M+1}$. Поскольку $\binom{n}{bn} M^{bn} < (M+1)^n$, левая часть неравенства 5 оценивается сверху как $\log_2(M+1)$. Тогда найдется такая Δ, что вне Δ-окресности b_{max} значение не будет превосходить $\log_2(M + 1 - \varepsilon)$. □

Данная лемма позволяет считать, что b отделено от нуля некоторой константой. Пусть фиксирована b, задающая bn - размер подмножества X. Выберем в графе максимальное паросочетание. Пусть оно содержит $(1-\alpha)n$ вершин, обозначим их W. Ясно, что $\alpha \leq \frac{i(S)}{n} \leq \lambda$. Также ясно, что можно считать, что α и λ достаточно близки к 1: при увеличении λ оценка на $i(S)$ только слабеет, а при увеличении α можно рассматривать W как меньшее множество. Фиксируем также $\beta = \frac{|X \setminus W|}{n}$. По

аналогичным соображениям можно считать β фиксированным и рассматривать только сумму

$$\sum_{X_1 \subset W:\, |X_1| = (b-\beta)n} \quad \sum_{X_2 \subset (S \setminus W):\, |X_2| = \beta \cdot n} M^{\mu|X| + (1-\mu)t(X)} \tag{6}$$

Тогда $t(X)$ можно оценить как $\beta n + \frac{b-\beta}{2} n = \frac{b+\beta}{2} n$, так как в W не более половины вершин могут образовывать независимое множество. Оценим сумму 6 как

$$\binom{\alpha n}{\beta n} \binom{(1-\alpha)n}{(b-\beta)n} M^{\mu b n + (1-\mu)\frac{b+\beta}{2} n} = \binom{\alpha n}{\beta n} \binom{(1-\alpha)n}{(b-\beta)n} M^{\left(\frac{1+\mu}{2} b + \frac{1-\mu}{2} \beta\right) n} \tag{7}$$

Достаточно оценить выражение 7 величиной $\mathcal{O}^* \left(\binom{n}{bn} M^{(1-\varepsilon(\lambda))bn} \right)$. Заметим, что оценки $\beta < b$ недостаточно. Из леммы 25 следует, что достаточно доказать только следующую оценку:

Лемма 28.

$$\alpha H\left(\frac{\beta}{\alpha}\right) + (1-\alpha) H\left(\frac{b-\beta}{1-\alpha}\right) + \log_2(M) \left(\frac{1+\mu}{2} b + \frac{1-\mu}{2} \beta\right) < \\ < H(b) + \log_2(M)(1-\varepsilon(\lambda))b \tag{8}$$

Лемма 29.

$$\alpha H\left(\frac{\beta}{\alpha}\right) + (1-\alpha) H\left(\frac{b-\beta}{1-\alpha}\right) < H(b)$$

.

Доказательство. Следует из леммы 25 и утверждения

$$\binom{\alpha n}{\beta n} \binom{(1-\alpha)n}{(b-\beta)n} < \binom{n}{bn}$$

□

Доказательство. (Леммы 28) Для того чтобы найти максимум левой части выражения 8, возьмем производную по β и приравняем к нулю:

$$\log_2\left(\frac{\alpha-\beta}{\beta}\right) - \log_2\left(\frac{1-\alpha-b+\beta}{b-\beta}\right) + \log_2(M)\frac{1-\mu}{2} = 0.$$

Обозначим

$$L = M^{\frac{1-\mu}{2}}, \;\; \delta = b - \beta.$$

Тогда имеем

$$\frac{1-\alpha-\delta}{\delta} = \frac{\alpha - b + \delta}{b-\delta} L.$$

$$\delta^2 + 2B\delta - C = 0,$$

где $B = \left(\alpha - b + \frac{1}{L-1}\right)/2$, $C = \frac{b(1-\alpha)}{L-1}$. Решением является

$$\delta = -B + B\sqrt{1 + \frac{C}{B^2}} = \frac{C}{2B} + \mathcal{O}\left(\frac{C^2}{B^3}\right).$$

То есть

$$\delta = K(1-\alpha)b + \mathcal{O}((1-\alpha)^2),$$
$$K = \frac{1}{2B(L-1)} = \frac{1}{(\alpha - b)\left(M^{\frac{1-\mu}{2}} - 1\right) + 1}.$$

Можно оценить K с учётом того, что $\alpha \approx 1$:

1. Если $M = 2$, $\mu = \frac{1}{2}$, то $K > 0.84$
2. Если $M = 3$, $\mu = \frac{\log_2(5)}{2} \approx 0.732$, то $K > 0.86$

Итак, $\beta_{max} = b - \delta = b - K(1-\alpha)b + \mathcal{O}((1-\alpha)^2) \approx b(1 - K(1-\alpha))$ при маленьких $1 - \alpha$.

Подставляя в левую часть неравенства 8 полученную β_{max}, видим, что

$$\left(\frac{1+\mu}{2}b + \frac{1-\mu}{2}\beta\right) < \left(1 - \frac{K(1-\mu)}{2}(1-\lambda)\right) b < (1 - \varepsilon(\lambda))b$$

С учётом леммы 29, завершаем доказательство неравенства 8. □

Теорему 23 можно сформулировать в более категоричной форме:

Теорема 30. Пусть дан граф G, представленный в виде обьединения S и $F \in \mathfrak{C}$. Пусть $\mathfrak{C}$ - класс графов, который допускает добавление изолированных вершин. Пусть также $SC(\mathfrak{C})$ можно решить за время

$\mathcal{O}^*\left(2^{|S_1|}\right)$, а задачу DS можно решить за время $\mathcal{O}^*\left((3-\varepsilon)^{|S|}\right)$ для некоторого $\varepsilon > 0$ при условии что S - независимое множество. Тогда для некоторого $\delta > 0$ DS можно решить за время $\mathcal{O}^*\left((3-\delta)^{|S|}\right)$, но без условия на независимость S.

Доказательство. Рассмотрим S, F, требуется найти минимальное доминирующее множество D. Найдём в S макимальное независимое множество. Пусть его размер $i(S) < \lambda|S|$ (число $\lambda < 1$ будет выбрана позже). Тогда применим теорему 23. Пусть I - независимое множество, и $|I| \geq \lambda|S|$. Обозначим $J = S \setminus I$. Будем перебирать возможные варианты D_1 для $D \cap J$. Множество D_1 продоминировало какие-то вершины из I, J, а также из F. Уберём все вершины $N_J[D_1]$. Добавим в модулятор вершину x, соединённую с $N_F(D_1)$, а в F добавим вершину y, соединённую с x. Аналогично, добавим в F вершину x', соединённую с $N_I(D_1)$, а в модулятор y', соединённую с x'. Тогда, в DS для получившегося графа обязаны лежать вершины x, x', так как y, y' должны быть продоминированы. Поэтому будут продоминированы и вершины из $N(D_1)$. В J остались непродоминированными вершины из $J' = J \setminus N_J[D_1]$. Поскольку мы не берём их в ответ, можно считать, что в J' нет рёбер. Часть вершин из J' должна быть продоминирована вершинами из I, другая часть - вершинами из F. Переберём все подмножества $J_1 \subset J'$. Пусть J_1 продоминировано F. Тогда можно считать, что между J_1 и I нет рёбер. Поскольку тогда $J_2 = J' \setminus J_1$ продоминировано I, можно считать, что между J_2 и F нет рёбер. Перенесём J_2 в F.

В итоге мы получили F', равное $F \cup J_2 \cup \{y, x'\}$, и независимое S', равное $I \cup J_1 \cup \{x, y'\}$. Будем искать в полученном графе наименьшее доминирующее множество D' за время $\mathcal{O}^*\left((3-\varepsilon)^{|S'|}\right)$, и выбирать наименьшее по всем вариантам D_1 и J_1 среди множеств $D' \cup D_1 \setminus \{x, x'\}$. Поймём, почему данный алгоритм работает корректно.

Пусть в полученном графе есть DS размера k. Тогда в исходном графе есть DS размера $k - 2 + |D_1|$, так как, убрав вершины x, x' и добавив D_1, мы получим доминирующее множество исходного графа необходимого размера (потенциально оно может содержать вершины из $J \setminus D_1$). Пусть требуется найти в исходном графе DS размера l. Тогда

по построению, в одном из вариантов для D_1, J_1 в полученном графе будет доминирующее множество размера $l - |D_1| + 2$.

Оценим теперь время работы алгоритма. Перебор идёт по всем $D_1 \subset J$ (напомним, что $|J| < (1-\lambda)|S|$), затем по всем J_1 (тоже самое, что по всем J_2):

$$\sum_{k=0}^{(1-\lambda)|S|} \binom{(1-\lambda)|S|}{k} \sum_{l=0}^{(1-\lambda)|S|-k} \binom{(1-\lambda)|S|-k}{l} \mathcal{O}^*\left((3-\varepsilon)^{(1-\lambda)|S|-k-l+2+\lambda|S|}\right) =$$

$$\mathcal{O}^*\left((3-\varepsilon)^{\lambda|S|}(5-\varepsilon)^{(1-\lambda)|S|}\right) = \mathcal{O}^*\left(\left((3-\varepsilon)^{\lambda}(5-\varepsilon)^{(1-\lambda)}\right)^{|S|}\right)$$

Для любого $\delta < \varepsilon$ можно выбрать такой λ, что выражение $(3-\varepsilon)^{\lambda}(5-\varepsilon)^{(1-\lambda)}$ будет меньше, чем $3 - \delta$. □

5.6. Выводы

- Была сформулирована общая схема алгоритмов решающих различные вариации DS, параметризованных расстоянием до классов графов.
- Был предложен алгоритм, параметризованный расстоянием d до кластерных графов, решающий PDS за время $\mathcal{O}^*\left(4^d\right)$.
- Были предложены алгоритмы, параметризованные расстоянием до интервальных графов, решающие DS и TDS за время $\mathcal{O}^*\left(3^d\right)$ и $\mathcal{O}^*\left(4^d\right)$ соответственно.
- Были предложены алгоритмы, параметризованные расстоянием до графов перестановок, решающие DS и TDS за время $\mathcal{O}^*\left(3^d\right)$ и $\mathcal{O}^*\left(4^d\right)$ соответственно.
- Был проведён анализ таких алгоритмов в зависимости от размера максимального независимого множества в модуляторе.

6. Заключение

В данной работе представлено множество результатов, связанных с вычислительной сложностью задачи Paired Dominating Set, а также задачи Dominating Set. Многие из них в разной степени сложности развивают существующие результаты, некоторые являются полностью самостоятельными результатами.

Для задачи Paired Dominating Set было получено несколько алгоритмов, среди которых:

- Точный экспоненциальный алгоритм, который имеет время работы $\mathcal{O}^*(1.7159^n)$ в общем случае. При наличии алгоритма для определённых классов графов, решающего задачу Dom-Enum за время $\mathcal{O}^*(\alpha^n)$, алгоритм для задачи PDS будет работать аналогичное время.
- Алгоритм, параметризованный шириной k древесной декомпозиции, имеющий время работы $\mathcal{O}^*\left(4^k\right)$.
- Алгоритм, параметризованный расстоянием d до кластерных графов, имеющий время работы $\mathcal{O}^*\left(4^d\right)$.

Кроме того, была доказана нижняя оценка для время работы алгоритма для PDS, параметризованного шириной древесной декомпозиции. А именно, в предположении SETH было показано, что $\forall \varepsilon > 0$ не существует такого алгоритма, работающего за время $\mathcal{O}^*\left((4-\varepsilon)^k\right)$.

Алгоритмы, параметризованные расстоянием до более сложных классов графов были получены для решения задач Dominating Set и Total Dominating Set. Была предложена общая схема таких алгоритмов, и были описаны алгоритмы для случая интервальных графов и графов перестановок. В обоих случаях время работы равно $\mathcal{O}^*\left(3^d\right)$ для задачи DS и $\mathcal{O}^*\left(4^d\right)$ для задачи TDS.

В попытках улучшить время работы таких алгоритмов, было показано, что если в модуляторе размер максимального независимого множества не превосходит фиксированной доли $\lambda < 1$ от всех вершин, то вышеописанные алгоритмы имеют время работы $\mathcal{O}^*\left((3-\varepsilon(\lambda))^d\right)$ и $\mathcal{O}^*\left((4-\varepsilon(\lambda))^d\right)$ соответственно.

Список литературы

[1] A. Sasireka A. H. Nandhu Kishore. Applications of Dominating Set of Graph in Computer Networks // nternational Journal of Engineering Sciences Research Technology. — 2014. — Access mode: https://www.sciencedirect.com/science/article/pii/019667748590001X.

[2] Booth Kellogg S., Lueker George S. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms // Journal of Computer and System Sciences. — 1976. — Vol. 13, no. 3. — P. 335–379. — Access mode: https://www.sciencedirect.com/science/article/pii/S0022000076800451.

[3] Chang Maw-Shang. Efficient Algorithms for the Domination Problems on Interval and Circular-Arc Graphs // SIAM J. Comput. — 1998. — Vol. 27. — P. 1671–1694.

[4] Chen Lei, Lu Changhong, Zeng Zhenbing. Hardness results and approximation algorithms for (weighted) paired-domination in graphs // Theoretical Computer Science. — 2009. — Vol. 410, no. 47. — P. 5063–5071. — Access mode: https://www.sciencedirect.com/science/article/pii/S0304397509005684.

[5] Chen Lei, Lu Changhong, Zeng Zhenbing. A linear-time algorithm for paired-domination problem in strongly chordal graphs // Information Processing Letters. — 2009. — Vol. 110, no. 1. — P. 20–23. — Access mode: https://www.sciencedirect.com/science/article/pii/S0020019009002828.

[6] Chen Lei, Lu Changhong, Zeng Zhenbing. Labelling Algorithms for Paired-domination Problems in Block and Interval Graphs // Journal of Combinatorial Optimization. — 2010. — 05. — Vol. 19.

[7] Combinatorial Bounds via Measure and Conquer: Bounding Minimal Dominating Sets and Applications / Fedor V. Fomin, Fabrizio Grandoni, Artem V. Pyatkin, Alexey A. Stepanov // ACM

Trans. Algorithms. — 2008. — Dec. — Vol. 5, no. 1. — Access mode: https://doi.org/10.1145/1435375.1435384.

[8] Enumeration and maximum number of minimal dominating sets for chordal graphs / Petr A. Golovach, Dieter Kratsch, Mathieu Liedloff, Mohamed Yosri Sayadi // Theor. Comput. Sci. — 2019. — Vol. 783. — P. 41–52. — Access mode: https://doi.org/10.1016/j.tcs.2019.03.017.

[9] Farber Martin, Mark Keil J. Domination in permutation graphs // Journal of Algorithms. — 1985. — Vol. 6, no. 3. — P. 309–321. — Access mode: https://www.sciencedirect.com/science/article/pii/019667748590001X.

[10] Goyal D. Jacob A. Kumar K. Majumdar D. Raman V. Structural Parameterizations of Dominating Set Variants. — 2018. — Access mode: https://doi.org/10.1007/978-3-319-90530-3_14.

[11] H. Qiao L. Kang M. Cardel, Du D. Z. Paired domination of trees. Dedicated to Professor J.B. Rosen on his 80th birthday // J. Global Optim. — 2003.

[12] Hassani Karbasi Amir, Ebrahimi Atani Reza. Application of dominating sets in wireless sensor networks // International Journal of Security and its Applications. — 2013. — 01. — Vol. 7. — P. 185–202.

[13] Haynes T. Slater P. Paired-domination in graphs. — 1998.

[14] Haynes T.W., Hedetniemi S., Slater P. Fundamentals of Domination in Graphs. Chapman & Hall/CRC Pure and Applied Mathematics. — Taylor & Francis, 1998. — ISBN: 9780824700331. — Access mode: https://books.google.ru/books?id=Bp9fot_HyL8C.

[15] Kang Liying, Sohn Moo Young, Cheng T.C.E. Paired-domination in inflated graphs // Theoretical Computer Science. — 2004. — Vol. 320, no. 2. — P. 485–494. — Access mode: https://www.sciencedirect.com/science/article/pii/S0304397504001276.

[16] Lappas Evaggelos, Nikolopoulos Stavros D., Palios Leonidas. An O(n)-Time Algorithm for the Paired-Domination Problem on Permutation Graphs // Combinatorial Algorithms / Ed. by Jiří Fiala, Jan Kratochvíl, Mirka Miller. — Berlin, Heidelberg : Springer Berlin Heidelberg, 2009. — P. 368–379.

[17] Lin Ching-Chi, Tu Hai-Lun. A linear-time algorithm for paired-domination on circular-arc graphs // Theoretical Computer Science. — 2015. — Vol. 591. — P. 99–105. — Access mode: https://www.sciencedirect.com/science/article/pii/S030439751500393X.

[18] Lokshtanov Daniel, Marx Dániel, Saurabh Saket. Known Algorithms on Graphs of Bounded Treewidth are Probably Optimal. — 2010. — 1007.5450.

[19] Minimal Dominating Sets in Graph Classes: Combinatorial Bounds and Enumeration / Jean-François Couturier, Pinar Heggernes, Pim van't Hof, Dieter Kratsch // SOFSEM 2012: Theory and Practice of Computer Science / Ed. by Mária Bieliková, Gerhard Friedrich, Georg Gottlob et al. — Berlin, Heidelberg : Springer Berlin Heidelberg, 2012. — P. 202–213.

[20] Misra N., Rathi Piyush. The Parameterized Complexity of Dominating Set and Friends Revisited for Structured Graphs // CSR. — 2019.

[21] Panda B.S., Pradhan D. A linear time algorithm for computing a minimum paired-dominating set of a convex bipartite graph // Discrete Applied Mathematics. — 2013. — Vol. 161, no. 12. — P. 1776–1783. — 9th Cologne/Twente Workshop on Graphs and Combinatorial Optimization (CTW 2010). Access mode: https://www.sciencedirect.com/science/article/pii/S0166218X12001734.

[22] Parameterized Algorithms / Marek Cygan, Fedor Fomin, Lukasz Kowalik et al. — 2015. — 01. — ISBN: 978-3-319-21274-6.

[23] Pradhan D., Panda B.S. Computing a minimum paired-dominating set in strongly orderable graphs // Discrete Applied Mathematics. —

2019. — Vol. 253. — P. 37–50. — 14th Cologne-Twente Workshop on Graphs and Combinatorial Optimization (CTW 2016). Access mode: https://www.sciencedirect.com/science/article/pii/S0166218X1830461X.

[24] Pătraşcu Mihai, Williams Ryan. On the Possibility of Faster SAT Algorithms // Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms. — SODA '10. — USA : Society for Industrial and Applied Mathematics, 2010. — P. 1065–1075.

[25] Rooij Johan, Bodlaender Hans. Design by Measure and Conquer, A Faster Exact Algorithm for Dominating Set. — Vol. 3404. — 2008. — 02. — P. 657–668.

[26] Spinrad Jeremy. Transitive Orientation in 0(n²) Time // Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing. — STOC '83. — New York, NY, USA : Association for Computing Machinery, 1983. — P. 457–466. — Access mode: https://doi.org/10.1145/800061.808777.

[27] Xu Yi-Zhi, Zhou Hai-Jun. Generalized minimum dominating set and application in automatic text summarization // Journal of Physics: Conference Series. — 2016. — 03. — Vol. 699. — P. 012014.

[28] van Rooij Johan M.M., Bodlaender Hans L. Exact algorithms for dominating set // Discrete Applied Mathematics. — 2011. — Vol. 159, no. 17. — P. 2147–2164. — Access mode: https://www.sciencedirect.com/science/article/pii/S0166218X11002393.

[29] van Rooij Johan M. M., Nederlof Jesper, van Dijk Thomas C. Inclusion/Exclusion Meets Measure and Conquer // Algorithms - ESA 2009 / Ed. by Amos Fiat, Peter Sanders. — Berlin, Heidelberg : Springer Berlin Heidelberg, 2009. — P. 554–565.