Санкт-Петербургский государственный университет

ВАГИН Евгений Юрьевич

Выпускная квалификационная работа

Тропические множества в графах

Образовательная программа бакалавриат «Математика»

Направление и код: 01.03.01 «Математика»

Шифр ОП: СВ.5000.2016

Научный руководитель:
доцент
Факультет математики
и компьютерных наук СПбГУ
кандидат ф.-м. наук
Близнец И.А.

Рецензент:
постдок
Гарвардский университет
кандидат ф.-м. наук
Головнёв А.Г.

Санкт-Петербург

2020 год

Оглавление

Глава 1

Введение

Разработка эффективных алгоритмов для перечисления всех возможных решений какой-то задачи восходит к 1950-м [1; 2; 3]. Цель перечисляющих алгоритмов – посчитать количество решений или вывести все решения одно за другим. Их изучение началось с задач сложности и оптимизации [4; 5; 6], затем распространилось на другие области, включая биоинформатику, машинное обучение, анализ сетей и социальный анализ [1; 7; 8] . В частности существует много задач на графах, ответ к которым - список подмножеств вершин, обладающих определённым свойством.

Одним из частных случаев задач, для которых разрабатываются перечисляющие алгоритмы – задача Графовые Узоры, введённая в 1994 Моррисом [9] и мотивированная различными приложениями, например, в биологии [10] и расчёте метаболических сетей [11]. Входом задачи является раскрашенный граф (G, c), где количество вершин – n, и M – какое-то мультимножество цветов графа, а c – функция раскраски, не обязательно правильная. Вопросом задачи ставится нахождение такого минимального связного индуцированного подграфа, что цвета вершин подграфа покрывают мультимножество M. В частности представляет интерес случай, когда мультимножество M определено как множество всех цветов исходного графа.

Такая задача называется Тропические Связные Множества, а

искомые подграфы называются *тропическими*. В работе [12] были получены алгоритмы со временем работы $\mathcal{O}(1.2721^n)$ для деревьев и $\mathcal{O}(1.5359^n)$ для произвольных графов, а также для задачи Тропические Связные Множества была показана NP-трудность даже для деревьев высоты 3.

В данной работе рассматриваются алгоритмы для задачи Перечисление Тропических Связных Множеств, в которой надо перечислить все минимальные тропические связные множества (далее MTCS). Эта задача впервые была изучена в статье [13]. В ней были получены нижние оценки количества MTCS и перечисляющие алгоритмы для некоторых видов графов. В частности для хордального графа была получена только нижняя оценка: был построен граф, в котором количество MTCS превышает 1.4916^n. Верхняя оценка на время работы перечисляющего алгоритма осталась тривиальной: $\mathcal{O}^*(2^n)$.

Хордальные графы определяются как графы, у которых любой цикл, имеющий четыре и более ребра, содержит хорду. Они играют важную роль во многих областях, таких как решение разреженных симметричных систем линейных уравнений [14] и системах управления базами данных [15].

Целью данной работы является построение алгоритмов, работающих быстрее, чем полный перебор. Мы представляем два алгоритма со временем работы $\mathcal{O}^*((2-\varepsilon)^n)$ для хордальных и произвольных графов соответственно.

Также был построен алгоритм для хордальных графов, параметризованный таким параметром как древесная ширина графа (treewidth). Этот параметр играет важную роль в разработке многих точных и приближённых алгоритмов для большого количества NP-трудных задач. Понятие treewidth было введено Робертсоном и Сеймуром [16] в своем доказательстве теоремы о минорах графа. Грубо говоря, древесная ширина измеряет похожесть графа на дерево. Например, древесная ширина дерева равна 1. У вышеупомянутого хордального графа с количеством MTCS, превышающем 1.4916^n, древесная ширина равна 8, поэтому на алгоритм, полиномиально зависящий

от treewidth, надеяться не приходится, но по крайней мере можно построить алгоритм, перечисляющий все MTCS с полиномиальной задержкой.

Результат данной работы

В данной работе получены следующие результаты.

Для задачи Перечисление Тропических Связных Множеств построены алгоритмы со временем работы

1. $\mathcal{O}^*(1.994^n)$ для хордальных графов;
2. $\mathcal{O}^*(1.9999993^n)$ для произвольных графов;
3. $\mathcal{O}^*(\max(\Delta, 2^{\mathrm{tw}} \cdot 4^C \cdot n))$ для хордальных графов с древесной шириной tw, количеством цветов C и количеством MTCS Δ.

Время работы первых двух построенных алгоритмов даёт верхнюю оценку на количество минимальных тропических связных множеств в произвольных и в хордальных графах. Время работы третьего алгоритма можно рассматривать так: алгоритм перечисляет все MTCS за $\mathcal{O}^*(2^{\mathrm{tw}} \cdot 4^C \cdot n)$, если их количество не превышает эту оценку, в противном случае алгоритм перечисляет все MTCS с полиномиальной задержкой.

Глава 2

Предварительные сведения

Обозначения и постановка задачи

Мы используем общепринятые обозначения теории графов. Будем рассматривать только простые графы, то есть неориентированные графы без петель и кратных ребер.

Для графа G, $V(G)$ – множество вершин G, $E(G)$ – множество рёбер графа G. Мы полагаем $n = |V(G)|$. $N(v)$ – множество соседей вершины $v \in V(G)$, $N[v] = N(v) \cup \{v\}$ – множество соседей вершины v, включая её саму. $\deg(v) = |N(v)|$ – степень вершины v в G. $G[F]$ – подграф G, порождённый подмножеством его вершин $F \subseteq V(G)$. *Кликой* называется такое подмножество вершин $D \subseteq V(G)$, такое что $G[D]$ – полный граф. $c : V(G) \to \mathbb{N}$ – функция раскраски (не обязательно правильная), сопоставляющая каждой вершине графа какой-то цвет. $c(X) = \{c(v) : v \in X\}$ – множество разных цветов, назначенных вершинам $X \subseteq V(G)$. Множество всех цветов графа – $\mathcal{C} = c(V(G))$. *Тропическое множество* графа G – такое подмножество его вершин X, что $c(X) = c(V(G))$. *Тропическое связное множество* графа G – такое подмножество его вершин X, что X – тропическое и $G[X]$ – связный подграф. Будем считать, что $\gamma := \frac{|\mathcal{C}|}{n}$. Также введём понятие *радужное множество* – это тропическое множество размера $|\mathcal{C}|$. Нетрудно заметить, что цвета всех вершин в таком множестве разные.

Хордальный граф – это такой граф, у которого любой цикл, имеющий четыре и более ребра, содержит хорду (то есть ребро, соединяющее две вершины цикла, но не являющееся его частью).

Древесная декомпозиция графа G – это такая пара $(\{X_i : i \in I\}, T)$, в которой каждое множество X_i для $i \in I$ – это подмножество вершин V, называемое *сумкой* и T - дерево с элементами I в качестве узлов, которое удовлетворяет следующим свойствам:

1. $\cup_{i \in I} X_i = V$
2. Для любого ребра $\{u, v\} \in E$ существует $i \in I$ такой что $\{u, v\} \subseteq X_i$
3. для любых $i, j, k \in I$, если j находится на пути из i в k в T, то $X_i \cap X_k \subseteq X_j$

Хорошая древесная декомпозиция графа G – это древесная декомпозиция $(T, \{X_t\}_{t \in V(T)})$, в которой T – корневое дерево с выделенным корневым узлом r и выполняются следующие свойства:

- $X_r = \varnothing$ и $X_l = \varnothing$ для любого листа l из T.
- Всякий нелистовой узел T имеет один из следующих типов:
    1. **Включающий узел**: узел t с одним ребёнком t', такой что $X_t = X_{t'} \cup \{v\}$ для какой-то вершины $v \notin X_{t'}$. Будем говорить, что v *включена* в t.
    2. **Забывающий узел**: узел t с одним ребёнком t', такой что $X_t = X_{t'} \setminus \{w\}$ для какой-то вершины $w \in X_{t'}$. Будем говорить, что w *забывается* в t.
    3. **Соединяющий узел**: узел t с двумя детьми t_1 и t_2, такими что $X_t = X_{t_1} = X_{t_2}$.

Ширина древесной декомпозиции равна $\max_{i \in I} |X_i| - 1$. *Древесной шириной* графа G называется минимальная древесная ширина по всем возможным древесным декомпозициям и обозначается как $\mathrm{tw}(G)$.

Нотация $\mathcal{O}^*$ опускает полиномиальный множитель. Таким образом запись $\mathcal{O}^*(T(x))$ для времени работы алгоритма означает $\mathcal{O}(T(x) \cdot \mathrm{poly}(x))$, где

$T(x)$ – некая функция от x, размера задачи.

Дадим формальное определение задачи перечисления минимальных тропических связных множеств графа:

Перечисление Тропических Связных Множеств	
Входные данные:	граф G с раскраской $c : V(G) \to \mathbb{N}$, множество цветов $\mathcal{C}$
Вопрос задачи:	Перечислить все $X \subseteq V(G)$ такие что $G[X]$ - тропическое связное множество минимального размера

В дальнейшем мы будем считать, что граф G связный. Если это не так, то запустим алгоритм на компонентах связности графа.

Доказанные ранее утверждения

Нам понадобится ряд уже известных ранее утверждений.

Лемма 1. *[17] Для* $\alpha \in (0, 1)$,

$$\binom{n}{k} = \mathcal{O}^*\Big(\theta(k/n)^n\Big) \tag{2.1}$$

где $\theta(\alpha) = (\frac{1}{\alpha})^\alpha \cdot (\frac{1}{1-\alpha})^{1-\alpha}$

Определение 1. *Для множества терминальных вершин* $T \subset V(G)$ *надмножество* $S \supseteq T$ *назовём* T***-связным множеством****, если* S *индуцирует связный подграф и* S *– минимально по включению, то есть никакое строго меньшее подмножество не является* T*-связным.*

Теорема 1. *[18] Для множества терминалов* $T \subseteq V(G) : |T| \leq n/3$ *существует не более чем* $\binom{n-|T|}{|T|-2} \cdot 3^{(n-|T|)/3}$ *минимальных по включению* T*-связных множеств, и они могут быть перечислены за* $\mathcal{O}^*(\binom{n-|T|}{|T|-2} \cdot 3^{(n-|T|)/3})$.

Определение 2. *Для связного графа G подмножество вершин $X \subseteq V(G)$ назовём* ***связным доминирующим множеством****, если X индуцирует связный подграф и $N[X] = V(G)$.*

Теорема 2. *[19] Хордальный граф на n вершинах имеет не более чем 1.4736^n минимальных связных доминирующих множеств, и для перечисления этих множеств существует алгоритм, работающий за $\mathcal{O}^*(1.4736^n)$.*

Теорема 3. *[12]. MTCS графа на n вершинах можно найти за $\mathcal{O}^*(1.5359^n)$.*

Определение 3. Вершинный разделитель *(далее* разделитель*) — это такой набор вершин, удаление которого делает оставшийся граф несвязным.*

Определение 4. Симплициальная вершина – *это такая вершина, соседи которой образуют клику.*

Лемма 2. *[20]. В хордальном графе присутствует по крайней мере одна симплициальная вершина.*

Лемма 3. *[21] Пусть $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ – древесная декомпозиция неполного графа G и пусть ab - ребро T. Тогда, если $X_a \nsubseteq X_b$ и $X_b \nsubseteq X_a$, $X_a \cap X_b$ – разделитель G.*

Лемма 4. *[17]. Если для графа G существует древесная композиция с древесной шириной не более чем k, тогда для него существует также хорошая древесная декомпозиция шириной не более чем k. Более того, по данной древесной декомпозиции $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ графа G с шириной не более чем k можно за полиномиальное время построить хорошую древесную декомпозицию с не более чем $\mathcal{O}(k|V(G)|)$ узлами.*

Дополнительные леммы

Лемма 5. *Для хордального графа G существует древесная декомпозиция, во всех сумках которой лежат клики. Более того, такую декомпозицию можно построить за полиномиальное время.*

Доказательство. Если граф полный, то просто кладём все вершины в одну сумку. Поэтому будем полагать, что граф не полный.

Докажем по индукции по n, то есть количеству вершин графа. База: граф из одной вершины v хордальный, древесная декомпозиция для него состоит из одной сумки $X_r = \{v\}$. Переход: по лемме в графе G существует симплициальная вершина v. Рассмотрим граф G', полученный удалением из графа G вершины v. По предположению индукции для него можно построить древесную декомпозицию $\mathcal{T}'$, во всех сумках которой лежат клики. Древесная декомпозиция $\mathcal{T}$ для G получается добавлением узла x, соответствующего сумке $C = N[v]$ к некому узлу t декомпозиции $\mathcal{T}'$, такому что мощность пересечения сумки этого узла с C максимальна. Если подходящих узлов несколько, выберем любой. Тогда для $\mathcal{T}$ выполняются свойства 1 и 2: граф G отличается от графа G' только наличием вершины v и рёбер, инцидентных v. Все они лежат в новой сумке $C = N[v]$.

Покажем, что выполняется свойство 3. Действительно, если в какой-то из сумок, отличных от X_t есть вершина $u \notin X_t$, то из леммы 3 следует, что u и v находятся в разных компонентах связности графа $G \setminus X_t$. А значит в сумках $\mathcal{T}'$, отличных от X_t, не может быть вершин из $C \setminus X_t$.

Таким образом построение древесной декомпозиции сводится к поиску симплициальных вершин и удалению вершин из графа. Все эти операции реализуемы за полиномиальное время, их количество $n - 1$, следовательно весь алгоритм работает за полиномиальное время. □

На рисунке 2.1 представлен пример хордального графа и древесной декомпозиции для него.

Заметим, что при модификации древесной декомпозиции в хорошую, свойство в сумках лежат клики не потеряется: в любой сумке хорошей древесной декомпозиции лежит подмножество какой-то сумки из исходной древесной декомпозии, а подмножество клики – клика.

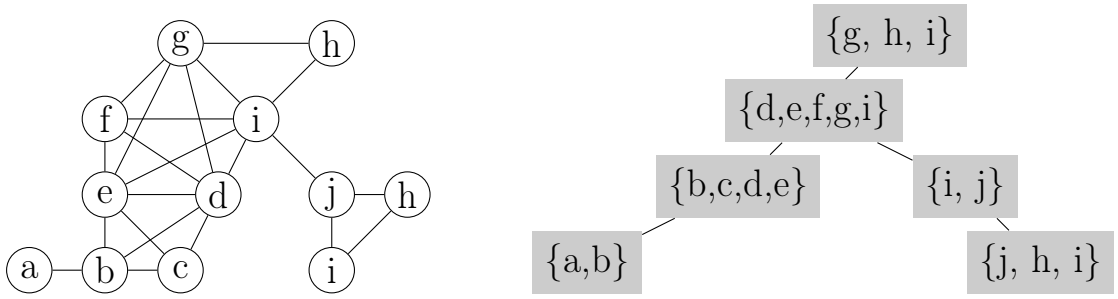

Рис. 2.1: Хордальный граф и его древесная декомпозиция

Утверждение 1. *Для хордального графа G существует хорошая древесная декомпозиция, во всех сумках которой лежат клики. Более того, такую декомпозицию можно построить за полиномиальное время.*

Лемма 6. *Для графа G на n вершинах, покрашенного в $C = |\mathcal{C}| = \gamma n$ цветов, количество радужных множеств не превышает $\mathcal{O}^*((\frac{1}{\gamma})^{\gamma n})$.*

Доказательство. Пусть $t_1, \ldots, t_C$ – количества вершин разных цветов. Тогда количество различных способов перебрать объединение непустых множеств верших одного цвета составляет $g(t_1, \ldots, t_C) = \prod_{i=1}^{C} t_i$. Заметим, что функция g достигает максимума на векторе $A = \{\frac{n}{C}, \ldots, \frac{n}{C}\}$. Пусть не так. Тогда существует вектор B, отличный от A, на котором достигается максимум. Рассмотрим вектор D, такой же как вектор B, но отличный в координатах $i, j : D_i = D_j = \frac{B_i + B_j}{2}$. Тогда $\frac{g(B)}{g(D)} = \frac{B_i \cdot B_j}{(\frac{B_i+B_j}{2})^2} = \left(\frac{\sqrt{B_i \cdot B_j}}{\frac{B_i+B_j}{2}}\right)^2 < 1$, так как для неравных аргументов среднее геометрическое меньше среднего арифметического. Откуда $g(B) < g(D)$, что противоречит исходному предположению. Значит максимум достигается на векторе A. □

Лемма 7. *Для графа G на n вершинах, покрашенного в $C = |\mathcal{C}| = \gamma n$ цветов, количество тропических множеств не превышает $\mathcal{O}^*((2^{1/\gamma} - 1)^{\gamma n})$.*

Доказательство. Пусть $t_1, \ldots, t_C$ – количества вершин разных цветов. Есть $2^{t_i} - 1$ подмножеств, содержащих вершины цвета i. Следовательно количество тропических множеств для графа G составляет $f(t_1, \ldots, t_C) = (2^{t_1} -$

$1) \cdots (2^{t_C} - 1)$. Докажем, что максимум функции f достигается на векторе $A = \{\frac{n}{C}, \ldots, \frac{n}{C}\}$. Действительно, пусть не так. Тогда максимум достигается на каком-то векторе B, отличном от A. Тогда у B существуют координаты $i, j : B_i \neq B_j$. Рассмотрим вектор D, такой же как вектор B, но отличный в координатах $i, j : D_i = D_j = \frac{B_i + B_j}{2}$. Тогда $\frac{f(B)}{f(D)} = \frac{\varphi(B_i)\varphi(B_j)}{\varphi^2(\frac{B_i+B_j}{2})}$, где $\varphi(x) = (2^x - 1)$. Заметим, что $\Phi = \ln \varphi$ - выпуклая вверх функция, поэтому по неравенству Йенсена получаем, что $\frac{\Phi(B_i)+\Phi(B_j)}{2} < \Phi(\frac{B_i+B_j}{2})$, откуда $\varphi(B_i)\varphi(B_j) < \varphi^2(\frac{B_i+B_j}{2})$, откуда $f(B) < f(D)$, что противоречит исходному предположению. Значит максимум достигается на векторе A. $\square$

Глава 3

Алгоритм для перечисления всех тропических связных множеств в произвольном графе

В этой главе представлен алгоритм для задачи Перечисление Тропических Связных Множеств для произвольных графов.

Алгоритм для большого количества цветов в графе

Теорема 4. *Для раскрашенного графа G на n вершинах, покрашенного в $C = |\mathcal{C}| = \gamma n$ цветов, существует алгоритм, который выводит все MTCS за $\mathcal{O}^*((2^{1/\gamma} - 1)^{\gamma n})$.*

Доказательство. Рассмотрим алгоритм, который перебирает все тропические множества и проверяет, является ли полученное подмножество минимальным и связным. Псевдокод алгоритма представлен на Рис. 3.1. По доказанной лемме 7 количество тропических множеств не превышает $\mathcal{O}^*((2^{1/\gamma} -$

$1)^{\gamma n}$). Откуда получаем оценку, предложенную в теореме. Связность множества легко проверяется, а для проверки минимальности надо знать размер минимального связного множества. Узнать его можно двумя способами:

1. Запустить вышеописанный алгоритм, поменяв его таким образом, чтобы он вместо вывода минимальных тропических связных множеств запоминал минимальный размер тропического связного множества. После этого можно запускать сам алгоритм. На асимптотику это не повлияет.
2. Запустить алгоритм из теоремы 3 для поиска минимального связного тропического множества. Для этого нужно, чтобы время работы алгоритма для поиска не превышало времени работы алгоритма для перечисления, то есть чтобы выполнялось $(2^{1/\gamma} - 1)^{\gamma} < 1.5359$. Это верно при $\gamma < 0.6395$. Если это ограничение не выполняется, доступен только первый способ.

□

Алгоритм: `minimal_mtcs`(G)

Вход: G – граф

Вывод: Все минимальные тропические связные множества $X \supseteq S$

для каждого $R_1 \subseteq \{S \in V(G) : c(S) = 1\}$ **выполнить**
  для каждого $R_2 \subseteq \{S \in V(G) : c(S) = 2\}$ **выполнить**
    ... **для каждого** $R_C \subseteq \{S \in V(G) : c(S) = C\}$ **выполнить**
      $S \leftarrow R_1 \cup R_2 \cup \cdots \cup R_C$
      если S */является MTCS/* **то**
        вывести S

Рис. 3.1: Алгоритм для вывода минимальных тропических связных множеств

Алгоритм для малого количества цветов в графе

Теорема 5. *Для графа G на n вершинах, покрашенного в $C = |\mathcal{C}| = \gamma n$ цветов, где $\gamma \leq 1/3$, существует алгоритм, который выводит все MTCS за $\mathcal{O}^*(((\frac{1}{\gamma})^\gamma \cdot \theta(\frac{\gamma}{1-\gamma}) \cdot (3^{\frac{1-\gamma}{3}}))^n)$, где $\theta(\alpha) = (\frac{1}{\alpha})^\alpha \cdot (\frac{1}{1-\alpha})^{1-\alpha}$*

Доказательство. Рассмотрим алгоритм, который для каждого радужного множества вершин $T \subseteq V(G) : |T| = \gamma n$, ищет все возможные T-связные множества S, после чего выводит множество $S \cup T$, если оно является минимальным. По лемме 6 количество таких множеств T не превышает $\mathcal{O}((\frac{1}{\gamma})^{\gamma n})$. По теореме 1 все такие такие множества S можно перечислить за $\mathcal{O}^*(\binom{n-|T|}{|T|-2} \cdot 3^{(n-|T|)/3}) = \mathcal{O}^*(\binom{n-\gamma n}{\gamma n-2} \cdot 3^{(n-\gamma n)/3})$. По лемме 1 биномиальный коэффициент $\binom{n-\gamma n}{\gamma n-2}$ оценивается как $\mathcal{O}^*(\theta(\frac{\gamma n-2}{n-\gamma n})) = \mathcal{O}^*(\theta(\frac{\gamma}{1-\gamma})^n)$. Из чего имеем утверждение теоремы. □

Итоговый алгоритм

Теорема 6. *Для графа G на n вершинах, покрашенного в $C = |\mathcal{C}| = \gamma n$ цветов, существует алгоритм, который выводит все минимальные тропические связные множества за $\mathcal{O}^*((2-\varepsilon)^n)$, где $\varepsilon = \frac{7}{10^7}$.*

Доказательство. Алгоритм принимает на вход граф G, после чего запускает алгоритм из теоремы 5, если в графе небольшое количество цветов, в противном случае запускает алгоритм из теоремы 4. Найдём границу γ_1, на которой заканчивается небольшое количество цветов. Заметим, что время работы алгоритма из 5 монотонно растёт по γ, а время работы алгоритма из 4 монотонно убывает по γ. Поэтому γ_1 является решением уравнения $(\frac{1}{\gamma})^\gamma \cdot \theta(\frac{\gamma}{1-\gamma}) \cdot (3^{\frac{1-\gamma}{3}}) = (2^{1/\gamma} - 1)^\gamma$. Решив уравнение, находим, что $\gamma_1 = 0.0477996$, откуда итоговое время работы меньше $\mathcal{O}(1.9999993^n)$, что соответствует условию теоремы. $\gamma_1 < 1/3$, поэтому использование алгоритма из 5 корректно. На рисунке 3.2 представлены графики оснований для различных значений γ. □

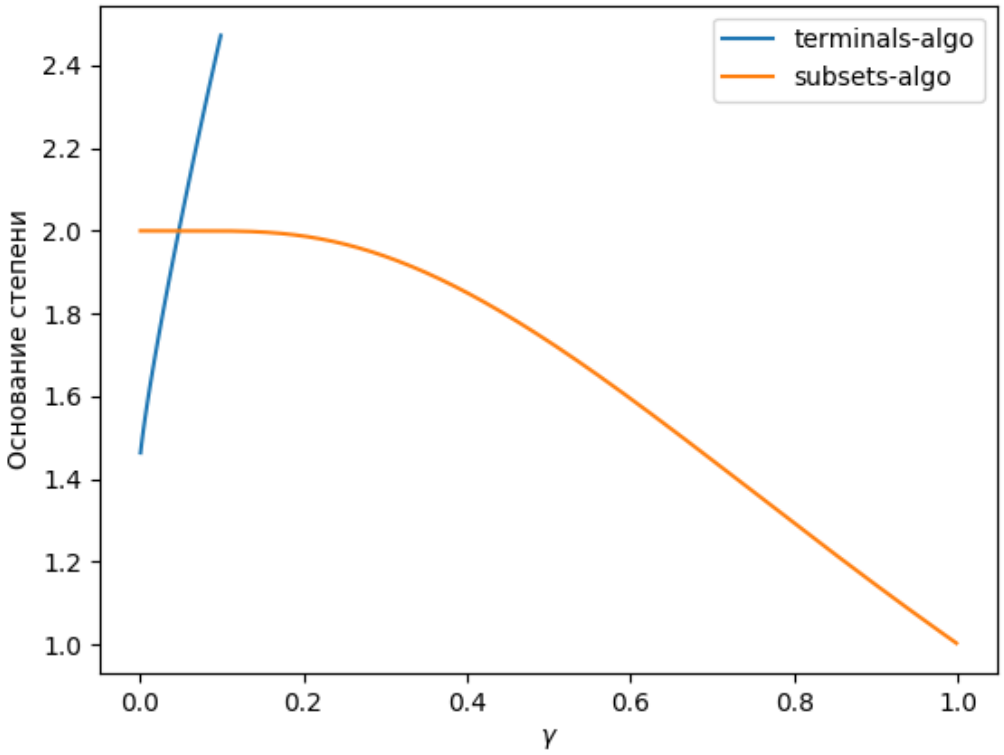

Рис. 3.2: Графики оснований в зависимости от γ

Глава 4

Алгоритм для перечисления всех тропических связных множеств в хордальном графе

Алгоритм для малого количества цветов в графе

Теорема 7. *Для хордального графа G на n вершинах, покрашенного в $C = |\mathcal{C}| = \gamma n$ цветов, существует алгоритм, который выводит все минимальные тропические связные множества за $\mathcal{O}^*((\frac{1}{\gamma})^\gamma \cdot 1.4736)^n)$.*

Для доказательства нам потребуется новое определение и лемма. Для подмножества вершин $X \subseteq V(G)$ какого-то хордального графа назовём *X-сужением* хордальный граф G_X полученный следующим образом:

1. Для графа G строим древесную декомпозицию $\mathcal{T}$, описанную в 1.
2. Находим минимальное по включению поддерево $\mathcal{T}_X$ (тоже являющееся древесной декомпозицией) для $\mathcal{T}$, содержащее все вершины из X.
3. Получаем граф G_X удалением из G вершин, не принадлежащих сумкам

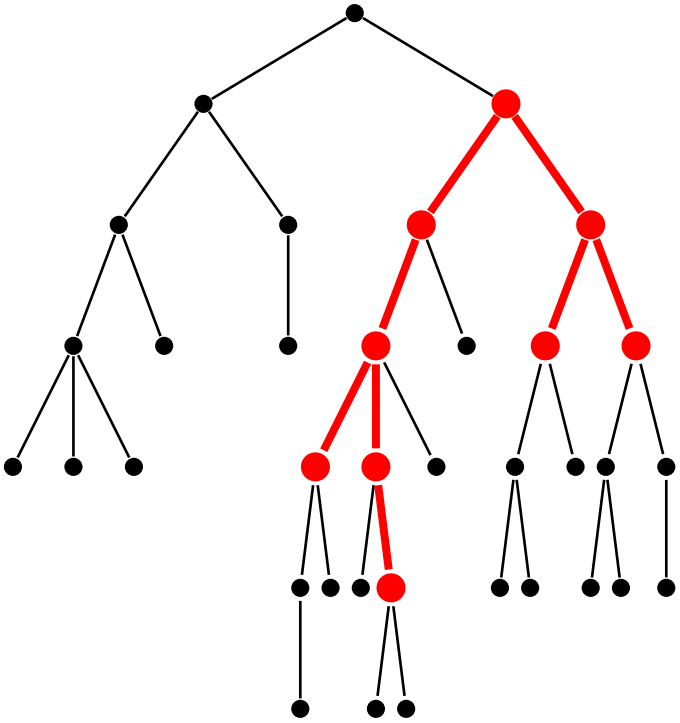

Рис. 4.1: древесная декомпозиция $\mathcal{T}$ и $\mathcal{T}_X$, последняя выделена красным

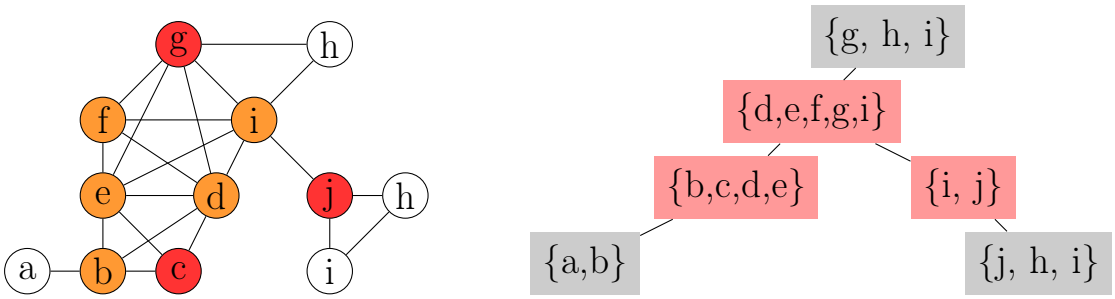

Рис. 4.2: Выбранные вершины и сумки, если $X = \{c, g, j\}$

$\mathcal{T}_X$.

На рисунке 4.1 представлена иллюстрация для $\mathcal{T}$ и $\mathcal{T}_X$. На рисунке 4.2 представлен пример хордального графа, его древесной декомпозиции и поддерева с выбранными вершинами X.

Лемма 8. *Любое связное множество S, содержащее $X \subseteq V(G)$, является доминирующим множеством в X-сужении графа G.*

Доказательство. Рассмотрим $\mathcal{T}_X$ и произвольное связное множество $S \supseteq X$. Так как все сумки $\mathcal{T}_X$ являются разделителями, для обеспечения связности в

каждой из них должна присутствовать по крайней мере одна вершина. А так как все сумки $\mathcal{T}_X$ являются кликами, множество S является доминирующим. □

Теперь мы готовы перейти к доказательству теоремы.

Доказательство. Алгоритм перебирает все радужные множества T, для каждого находит граф G_T, являющийся T-сужением графа G, после чего запускает алгоритм из теоремы 2, перебирающий все связные доминирующие подмножества S в G_T и выводящий $M := S \cup T$, если M является MTCS. Будем считать, что древесная декомпозиция, используемая в построении T-сужения, строится один раз и заранее. При этом в построенной декомпозиции никакая сумка не является подмножеством другой.

Заметим, что все MTCS графа G будут выведены в процессе работы этого алгоритма: для любого M, являющегося MTCS графа G, найдётся радужное множество X, являющееся подмножеством MTCS: $X \subseteq M$. Все минимальные связывающие множества лежат в X-сужении графа G: G_T. Если бы это было не так, можно было бы удалить из связывающего множества все вершины, не принадлежащие G_T, при этом не потеряв в связности. Теперь достаточно заметить, что так как по лемме 8 любое связное множество на графе G_T является доминирующим, алгоритм из 2 переберёт в том числе и M.

Осталось посчитать время работы алгоритма. По лемме 6 количество радужных множеств T не превышает $(\frac{1}{\gamma})^{\gamma n}$. Размер графа G_T, построенного для каждого T оценивается как размер исходного графа n, таким образом время работы алгоритма для поиска связных доминирующих множеств не превышает $\mathcal{O}^*(1.4736^n)$, и таким образом получается заявленная асимптотика: $\mathcal{O}^*(((\frac{1}{\gamma})^{\gamma} \cdot 1.4736)^n)$ □

Итоговый алгоритм

Теорема 8. *Для раскрашенного хордального графа G на n вершинах, существует алгоритм, который выводит все минимальные тропические связные множества за* $\mathcal{O}^*(1.994^n)$

Доказательство. Алгоритм строится аналогично итоговому алгоритму из главы для произвольных графов: если в графе небольшое количество цветов, запускаем алгоритм из 7, в противном случае запускаем алгоритм из теоремы 4. Время работы монотонно по γ, поэтому граница аналогично находится решением уравнения $(\frac{1}{\gamma})^\gamma \cdot 1.4736 = (2^{1/\gamma} - 1)^\gamma$. Решив уравнение находим, что $\gamma_1 = 0.171518 \rightarrow$, откуда итогове время работы $\mathcal{O}^*(1.994^n)$. На рисунке 4.3 представлены графики оснований для различных значений γ. □

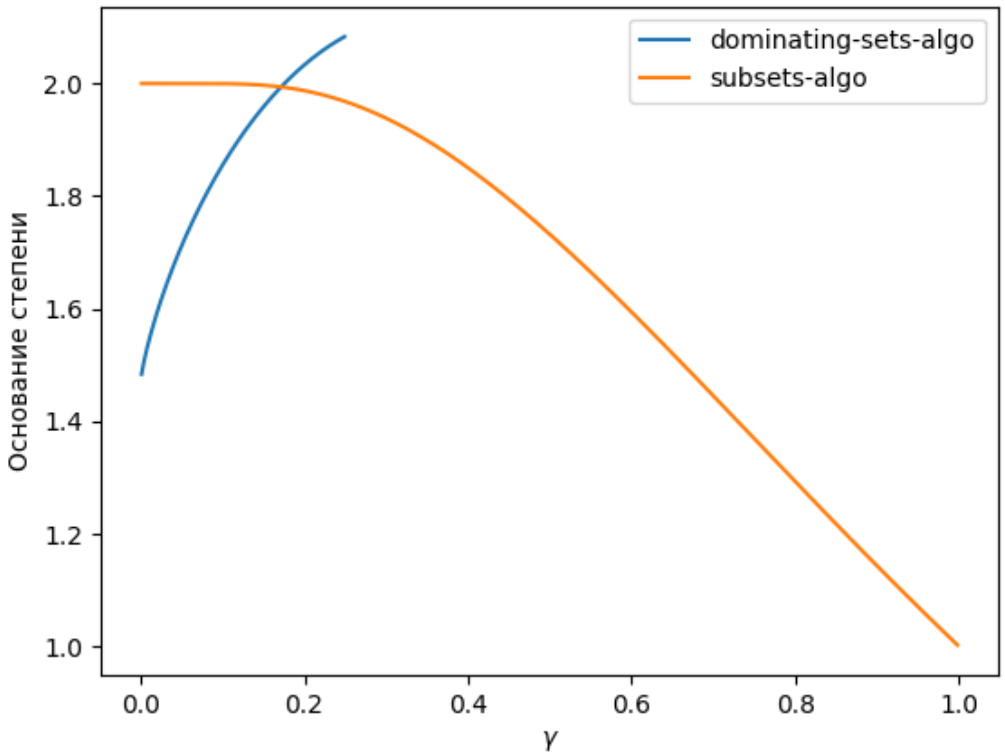

Рис. 4.3: Графики оснований в зависимости от γ

Глава 5

Алгоритм для хордальных графов с небольшой древесной шириной

Описание алгоритма

Пусть на вход дан раскрашенный хордальный граф (G, c) с количеством вершин $|G| = n$ и множеством цветов $\mathcal{C}$. Опишем алгоритм, работающий на основе древесной декомпозиции. По утверждению 1 мы можем считать, что для графа G построена хорошая древесная декомпозиция $(\{X_i : i \in I\}, T)$ с древесной шириной tw, причём во всех сумках лежат клики, являющиеся разделителями. Основная идея состоит в том, что мы определим подзадачу нашей исходной задачи таким образом, чтобы её можно было легко хранить в конкретных узлах T.

Определим для каждого узла t, для каждого подмножества $S \subseteq X_t$ и каждого подмножества $\mathcal{P}_c \subseteq \mathcal{C}$ функцию $d(t, S, \mathcal{P}_c)$, показывающую, какое минимальное количество включающих S вершин нужно, чтобы покрыть множество цветов $\mathcal{P}_c$. Также для соединяющих узлов определим функцию $backtrace(t, S, \mathcal{P}_c)$, возвращающую множество пар $(\mathcal{P}_{c_1}, \mathcal{P}_{c_2})$ для t_1 и t_2 со-

ответственно, таких что $\mathcal{P}_c = \mathcal{P}_{c_1} \cup \mathcal{P}_{c_2}$, на которых достигается минимум функции $d(t, S, \mathcal{P}_c)$.

Тогда алгоритм будет состоять из двух шагов:

1. Динамическое вычисление значений функций d и $backtrace$ от листьев к корню.
2. Рекурсивное перечисление всех MTCS от корня к листьям с помощью функций d и $backtrace$.

В псевдокоде ниже введены 2 обозначения:

1. Оператор $\xleftarrow{\min}$ означает операцию "присвоить, если меньше"(запись $a \xleftarrow{\min} b$ эквивалента записи $a \leftarrow \min(a, b)$).
2. Оператор $\xleftarrow{+}$ означает операцию "добавить в множество"(запись $a \xleftarrow{+} b$ эквивалента записи $a \leftarrow a \cup \{b\}$).

Шаг 1: вычисление значений

Листовой узел. Если t – листовой узел, то $d(t, \varnothing, \varnothing) = 0$

Включающий узел. Пусть t - включающий узел с ребёнком t', такой что $X_t = X_{t'} \cup \{v\}$ для какого-то $v \notin X_{t'}$. Тогда для любого S, такого что $S \subseteq X_t$, а $\mathcal{P}_c \subseteq \mathcal{C}$ верна формула:

$$d(t, S, \mathcal{P}_c) = \begin{cases} d(t', S \setminus \{v\}, \mathcal{P}_c \cup \{c(v)\}) + 1, & \text{если } v \in S \\ d(t', S, \mathcal{P}_c), & \text{если } v \notin S \end{cases}$$

Соединяющий узел. Пусть t - соединяющий узел с детьми t_1 и t_2, такой что $X_t = X_{t_1} = X_{t_2}$. Пусть S – подмножество X_t. Определим на $\mathcal{C}^2$ функцию $\omega(\mathcal{P}_{c_1}, \mathcal{P}_{c_2}) = d(t_1, S, \mathcal{P}_{c_1}) + d(t_2, S, \mathcal{P}_{c_2})$, показывающую, какое минимальное количество вершин нужно в поддеревьях $\mathcal{T}_{t_1}$ и $\mathcal{T}_{t_2}$, чтобы покрыть цвета $\mathcal{P}_{c_1}$ и $\mathcal{P}_{c_2}$ соответственно. Тогда для любого множества цветов $\mathcal{P}_c \subseteq \mathcal{C}$ определим наши функции как $d(t, S, \mathcal{P}_c) := \min\limits_{\mathcal{P}_{c_1} \cup \mathcal{P}_{c_2} = \mathcal{P}_c} w(\mathcal{P}_{c_1}, \mathcal{P}_{c_2}) - |S|$, и $backtrace(t, S, \mathcal{P}_c) := \operatorname{argmin}\limits_{\mathcal{P}_{c_1} \cup \mathcal{P}_{c_2} = \mathcal{P}_c} w(\mathcal{P}_{c_1}, \mathcal{P}_{c_2})$ (напомним, что значение $backtrace$

– множество пар, то есть минимум может достигаться более чем на одной паре множеств цветов).

Забывающий узел. Пусть t – забывающий узел с ребёнком t', такой что $X_t = X_{t'} \setminus \{w\}$ для какого-то $w \in X_{t'}$. Пусть S - подмножество X_t, а $\mathcal{P}_c$ подмножество цветов C. Тогда, как и в алгоритме для включающего узла, верна похожая формула:

$$d(t, S, \mathcal{P}_c) = \begin{cases} \min(d(t', S, \mathcal{P}_c), d(t', S \cup \{w\}, \mathcal{P}_c)), & \text{если } v \in S \\ d(t', S, \mathcal{P}_c), & \text{если } v \notin S \end{cases}$$

Теорема 9. *Время работы алгоритма в* ***первом шаге*** *составляет* $\mathcal{O}^*(2^{\mathrm{tw}} \cdot 4^C \cdot n)$*, где* n *- количество вершин,* tw *- древесная ширина графа, а* C *- количество цветов.*

Доказательство. Вычисление функций d и *backtrace* требует рассмотрения всех возможных состояний, посчитаем их количество в различных узлах:

1. Во включающем узле $|\{S : S \subseteq X_t\}| \times |\{\mathcal{P}_c : \mathcal{P}_c \subseteq \mathcal{C}\}| \leq 2^{\mathrm{tw}+1} \cdot 2^C$.
2. В соединяющем узле $|\{S : S \subseteq X_t\}| \times |\{\mathcal{P}_{c_1} : \mathcal{P}_{c_1} \subseteq \mathcal{C}\}| \times |\{\mathcal{P}_{c_2} : \mathcal{P}_{c_2} \subseteq \mathcal{C}\}| \leq 2^{\mathrm{tw}+1} \cdot 4^C$.
3. В забывающем узле, как и во включающем узле: не более $2^{\mathrm{tw}+1} \cdot 2^C$.

В древесной декомпозиции графа $\mathcal{O}(n)$ узлов, а в узле в худшем случае $2^{\mathrm{tw}+1} \cdot 4^C$ состояний, и таким образом получается заявленная асимптотика: $\mathcal{O}^*(2^{\mathrm{tw}} \cdot 4^C \cdot n)$ □

Шаг 2: Вывод всех MTCS

Вывод MTCS происходит с помощью процедуры `iterate_mtcs`$(t, S, \mathcal{P}_c)$, которая для узла t, подмножества вершин этого узла $S \subseteq X_t$ и подмножества цветов $\mathcal{P}_c \subseteq \mathcal{C}$ возвращает минимальные по размеру связные множества вершин Z, включающие S и покрывающие множество цветов $\mathcal{P}_c$. Также будем полагать, что если мы посчитали в каком-то узле t для каких-то параметров S и $\mathcal{P}_c$

значение `iterate_mtcs`, то мы запоминаем это значение на случай, если оно понадобится снова. Процедура запускается из корня r: $\texttt{iterate\_mtcs}(r, \varnothing, \mathcal{C})$.

Забывающий узел. Пусть t - забывающий узел с ребёнком t', такой что $X_t = X_{t'} \setminus \{w\}$ для какого-то $w \in X_{t'}$. Тогда мы с помощью функции d, посчитанной на первом шаге, проверяем, существуют ли решения для случаев если w принадлежит какому-то MTCS и если не принадлежит. Если существуют, перечисляем их. На рисунке 5.1 представлен псевдокод.

Алгоритм: $\texttt{iterate\_mtcs}(t, S, \mathcal{P}_c)$

$m \leftarrow d(t, S, \mathcal{P}_c)$

$M \leftarrow \varnothing$

если $d(t', S, \mathcal{P}_c) = m$ **то**

  $M \xleftarrow{+} \texttt{iterate\_mtcs}(t', S, \mathcal{P}_c)$

если $d(t', S \cup \{w\}, \mathcal{P}_c \setminus \{c(w)\}) = m - 1$ **то**

  $M \xleftarrow{+} \texttt{iterate\_mtcs}(t', S \cup \{w\}, \mathcal{P}_c \setminus \{c(w)\})$

вернуть M

Рис. 5.1: Перечисление MTCS в забывающем узле

Соединяющий узел. Пусть t - соединяющий узел с детьми t_1 и t_2, такой что $X_t = X_{t_1} = X_{t_2}$. Тогда возвращаем решения для множеств пар, на которых достигается минимум: $(\mathcal{P}_{c_1}, \mathcal{P}_{c_2}) \in backtrace(t, S, \mathcal{P}_c)$. На рисунке 5.2 представлен псевдокод.

Включающий узел. Пусть t - включающий узел с ребёнком t', такой что $X_t = X_{t'} \cup \{v\}$ для какого-то $v \notin X_{t'}$. В этом случае мы возвращаем все решения для узла t', при этом, если $v \in S$, добавляем в каждое решение v. На рисунке 5.3 представлен псевдокод.

Листовой узел. В этом узле всегда возвращаем множество с пустым элементом элементом ($\{\varnothing\}$).

Утверждение 2. *Время работы алгоритма во* ***втором шаге*** *составляет*

Алгоритм: `iterate_mtcs`$(t, S, \mathcal{P}_c)$

$M \leftarrow \varnothing$

для каждого $(\mathcal{P}_{c_1}, \mathcal{P}_{c_2}) \in backtrace(t, S, \mathcal{P}_c)$ **выполнить**

- $A \leftarrow$ `iterate_mtcs`$(t_1, S, \mathcal{P}_{c_1})$
- $B \leftarrow$ `iterate_mtcs`$(t_1, S, \mathcal{P}_{c_2})$
- $M \xleftarrow{+} \{a \cup b \mid a \in A, b \in B\}$

вернуть M

Рис. 5.2: Перечисление MTCS в соединяющем узле

Алгоритм: `iterate_mtcs`$(t, S, \mathcal{P}_c)$

если $v \in S$ **то**

- **вернуть** $\{p + \{v\} \mid p \in$ `iterate_mtcs`$(t', S \setminus \{v\}, \mathcal{P}_c)\}$

иначе

- **вернуть** `iterate_mtcs`$(t', S, \mathcal{P}_c)$

Рис. 5.3: Перечисление MTCS во включающем узле

$\mathcal{O}^*(\Delta)$, *где* Δ – *количество MTCS.*

Доказательство. По построению функций d и *backtrace*, процедура `iterate_mtcs` рекурсивно запускается из дочерних узлов, только если в них существуют какие-то решения. При этом больше никаких дополнительных вычислений мы не делаем. Поэтому время работы алгоритма оценивается количеством решений, то есть количеством MTCS. Откуда и получается заявленная оценка: $\mathcal{O}^*(\Delta)$ □

Расчёт итогового времени работы

Итоговое время работы – максимум по времени работы алгоритма в обоих шагах. Итого получается $\mathcal{O}^*(\max(\Delta, 2^{\mathrm{tw}} \cdot 4^C \cdot n))$, где Δ – количество MTCS.

Список использованных источников

[1] Efficient Graphlet Counting for Large Networks / Nesreen K. Ahmed, Jennifer Neville, Ryan A. Rossi, Nick Duffield // 2015 IEEE International Conference on Data Mining. — 2015. — 11. — P. nil. — URL: https://doi.org/10.1109/icdm.2015.141.

[2] Trent H. M. A Note on the Enumeration and Listing of All Possible Trees in a Connected Linear Graph // Proceedings of the National Academy of Sciences. — 1954. — Vol. 40, no. 10. — P. 1004–1007. — URL: https://doi.org/10.1073/pnas.40.10.1004.

[3] Gilbert E. N. Enumeration of Labelled Graphs // Canadian Journal of Mathematics. — 1956. — Vol. 8, no. nil. — P. 405–411. — URL: https://doi.org/10.4153/cjm-1956-046-2.

[4] Fukuda Komei. Note on new complexity classes ENP, EP and CEP.

[5] Itai Alon, Rodeh Michael. Finding a minimum circuit in a graph // Proceedings of the ninth annual ACM symposium on Theory of computing - STOC '77. — 1977. — -. — P. nil. — URL: https://doi.org/10.1145/800105.803390.

[6] Valiant L.G. The Complexity of Computing the Permanent // Theoretical

Computer Science. — 1979. — Vol. 8, no. 2. — P. 189–201. — URL: https://doi.org/10.1016/0304-3975(79)90044-6.

[7] Milo R. Network Motifs: Simple Building Blocks of Complex Networks // Science. — 2002. — Vol. 298, no. 5594. — P. 824–827. — URL: https://doi.org/10.1126/science.298.5594.824.

[8] Efficient graphlet kernels for large graph comparison / Nino Shervashidze, SVN Vishwanathan, Tobias Petri et al. // Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics / Ed. by David van Dyk, Max Welling. — Vol. 5 of Proceedings of Machine Learning Research. — Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA : PMLR, 2009. — 16–18 Apr. — P. 488–495. — URL: http://proceedings.mlr.press/v5/shervashidze09a.html.

[9] McMorris F. R., Warnow Tandy, Wimer Thomas. Triangulating vertex colored graphs // Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms. — Publ by ACM, 1993. — P. 120–127. — Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms ; Conference date: 25-01-1993 Through 27-01-1993.

[10] Koutis Ioannis. Constrained Multilinear Detection for Faster Functional Motif Discovery // Information Processing Letters. — 2012. — Vol. 112, no. 22. — P. 889–892. — URL: https://doi.org/10.1016/j.ipl.2012.08.008.

[11] Efficient Algorithms for Detecting Signaling Pathways in Protein Interaction Networks / Jacob Scott, Trey Ideker, Richard M. Karp, Roded Sharan // Journal of Computational Biology. — 2006. — . — Vol. 13, no. 2. — P. 133–144. — URL: https://doi.org/10.1089/cmb.2006.13.133.

[12] Exact Exponential Algorithms To Find Tropical Connected Sets of Minimum Size / Mathieu Chapelle, Manfred Cochefert, Dieter Kratsch et al. // Theoretical Computer Science. — 2017. — Vol. 676, no. nil. — P. 33–41. — URL: https://doi.org/10.1016/j.tcs.2017.03.003.

[13] Kratsch Dieter, Liedloff Mathieu, Sayadi Mohamed Yosri. Enumerating Minimal Tropical Connected Sets // SOFSEM 2017: Theory and Practice of Computer Science. — SOFSEM 2017: Theory and Practice of Computer Science. Springer International Publishing, 2017. — P. 217–228. — URL: https://doi.org/10.1007/978-3-319-51963-0_17.

[14] Rose Donald J. A GRAPH-THEORETIC STUDY OF THE NUMERICAL SOLUTION OF SPARSE POSITIVE DEFINITE SYSTEMS OF LINEAR EQUATIONS // Graph Theory and Computing. — Graph Theory and Computing. Elsevier, 1972. — P. 183–217. — URL: https://doi.org/10.1016/b978-1-4832-3187-7.50018-0.

[15] Tarjan Robert E., Yannakakis Mihalis. Simple Linear-Time Algorithms To Test Chordality of Graphs, Test Acyclicity of Hypergraphs, and Selectively Reduce Acyclic Hypergraphs // SIAM Journal on Computing. — 1984. — Vol. 13, no. 3. — P. 566–579. — URL: https://doi.org/10.1137/0213035.

[16] Robertson Neil, Seymour P.D. Graph Minors. Ii. Algorithmic Aspects of Tree-Width // Journal of Algorithms. — 1986. — Vol. 7, no. 3. — P. 309–322. — URL: https://doi.org/10.1016/0196-6774(86)90023-4.

[17] Fomin Fedor V., Kaski Petteri. Exact Exponential Algorithms // Communications of the ACM. — 2013. — Vol. 56, no. 3. — P. 80–88. — URL: https://doi.org/10.1145/2428556.2428575.

[18] Telle Jan Arne, Villanger Yngve. Connecting Terminals and 2-Disjoint Connected Subgraphs // Graph-Theoretic Concepts in Computer Sci-

ence. — Graph-Theoretic Concepts in Computer Science. Springer Berlin Heidelberg, 2013. — P. 418–428. — URL: https://doi.org/10.1007/978-3-642-45043-3_36.

[19] Enumeration of Minimal Connected Dominating Sets for Chordal Graphs / Petr A. Golovach, Pinar Heggernes, Dieter Kratsch, Reza Saei // Discrete Applied Mathematics. — 2020. — Vol. 278, no. nil. — P. 3–11. — URL: https://doi.org/10.1016/j.dam.2019.07.015.

[20] Dirac G. A. On Rigid Circuit Graphs // Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg. — 1961. — Vol. 25, no. 1-2. — P. 71–76. — URL: https://doi.org/10.1007/bf02992776.

[21] Parameterized Algorithms / Marek Cygan, Fedor V. Fomin, Łukasz Kowalik et al. []. — Springer International Publishing, 2015. — P. nil. — URL: https://doi.org/10.1007/978-3-319-21275-3.