Федеральное государственное автономное образовательное учреждение высшего образования
«Московский физико-технический институт
(национальный исследовательский институт)»

Физтех-школа прикладной математики и информатики
Кафедра дискретной математики

Направление: 01.03.02 Прикладные математика и информатика
Направленность (профиль): Прикладная математика и компьютерные науки

Выпускная квалификационная бакалаврская работа на тему:

Редукция данных в графической задаче коммивояжера

Студент _____ Скачков Д.А.
Научный руководитель Dr.rer.nat. _____ ван Беверн Р.А.
Зав. кафедрой д.ф-м.н., профессор _____ Райгородский А.М.

МОСКВА, 2021

Аннотация

В данной дипломной работе рассмотрена редукция данных для задачи коммивояжера на произвольном графе. Описана формальная постановка задачи, предложено два алгоритма построения полиномиальных ядер относительно таких параметров, как мощность наименьшего вершинного покрытия и мощность наименьшего разрезающего циклы набора ребер. Помимо этого в работе доказана невозможность существования ядра полиномиального размера относительно ряда других параметров, если только не выполнено вложение $\mathrm{coNP} \subseteq \mathrm{NP/poly}$ и не схлопывается полиномиальная иерархия.

Актуальность работы подтверждается тем, что задача коммивояжера является одной из фундаментальных задач маршрутизации, все известные алгоритмы решения которой имеют трудоемкость, не менее чем экспонененциальную по длине входа, в связи с чем алгоритмы редукции данных для этой задачи особенно важны. Практическая ценность работы, а также ее новизна, заключаются в том, что в ней впервые представлены разработанные нами алгоритмы с доказательством эффективности.

Содержание

1 Введение

Графическая задача коммивояжера состоит в том, чтобы построить как можно более короткий маршрут, проходящий через все заданные города, между которыми существуют дороги определенной длины. Входные данные представлены реберно-взвешенным графом, вершины которого сопоставлены городам, а ребра — дорогам. При всей простоте формулировки задача крайне трудна, и по некоторым причинам, о которых будет рассказано в дальнейшем, надежд на отыскание быстрых алгоритмов для ее решения не так уж много. Все известные на данный момент алгоритмы имеют не менее чем экспоненциальную трудоемкость числу вершин в графе. В связи с этим особый интерес представляют алгоритмы, удаляющие вершины графа так, чтобы решение исходной задачи можно было легко восстановить по решению новой, более простой.

Заметим, что если бы существовал полиномильный по числу вершин алгоритм, который для любого входа гарантировал бы удаление хотя бы одной вершины, сохраняя при этом решение, то такой алгоритм позволял бы и исходную задачу решать за полиномиальное время. Поэтому все известные полиномиальные алгоритмы редукции данных сжимают, во-первых, не любой вход, а во-вторых, лишь до определенной степени. Чтобы понимать, какие алгоритмы эффективны в каких случаях и насколько, каждому графу сопоставляется некоторый параметр, относительно которого и доказывается размер ядра, то есть доказывается, что алгоритм возвращает граф, число вершин которого является некоторой вычислимой функцией от значения введенного параметра.

В данной работе рассматривается существование ядер, размер которых ограничен полиномиальной функцией от ряда параметров, популярных в работах о построении ядер для задач на графах. Итоги исследования представлены на рисунке 1. Формальные определения тех параметров, которые непосредственно будут задействованы в данной работе, приводятся в дальнейшем.

Рис. 1.1: ([1]) *Желтым цветом обозначены те параметры, относительно которых существование ядер полиномиального размера означало бы схлопывание полиномиальной иерархии, красным — те, относительно которых существование ядер полиномиального размера означало бы равенство классов* ***P*** *и* ***NP****, зеленым — те, для которых доказано существование полиномиального ядра, про остальные же параметры ничего выяснить не удалось. Связь параметров* x *и* y *ребром обозначает* $x \leqslant poly(y)$ *в том случае, если* x *располагается ниже в иерархии.*

2 Основные определения и обозначения

2.1 Сложность вычислений

Напомним некоторые определения из теории сложности вычислений.

Определение 2.1.

- **DTIME**$(T(n))$ – класс языков, распознаваемых на детерминированной машине Тьюринга за время $O(T(n))$,
- **NTIME**$(T(n))$ – класс языков, распознаваемых на недетерминированной машине Тьюринга за время $O(T(n))$.

Определение 2.2.

- **P** $= \bigcup\limits_{k=1}^{\infty}$ **DTIME**(n^k),
- **NP** $= \bigcup\limits_{k=1}^{\infty}$ **NTIME**(n^k).

Поскольку детерминированная машина Тьюринга является частным случаем недетерминированной, вложение **P** $\subseteq$ **NP** очевидно. Строгость этого вложения, тем не менее, остается открытой проблемой современной математики.

Определение 2.3. Язык $L \subseteq \Sigma^*$ называется *NP-трудным*, если для любого другого языка $L' \in$ **NP** существует полиномиальный алгоритм, который слову W' сопоставляет W, так что $W \in L \iff W' \in L'$.
NP-полный язык по определению удовлетворяет двум условиям:

- лежит в классе **NP**,
- является NP-трудным.

Принадлежность любого NP-трудного языка классу **P** автоматически влечет за собой равенство классов **P** и **NP**. Поскольку ни для одной из NP-трудных задач полиномиальных алгоритмов до сих пор найдено не было, приходится мириться с тем, что гарантировать для них быстрое решение пока не представляется возможным.

2.2 Теория графов

Напомним также некоторые определения из теории графов. Мы будем использовать обозначение $S^{(k)}$ для множества всех k-элементных подмножеств множества S. Более формально:

$$S^{(k)} = \{R \subseteq S \mid |R| = k\}.$$

Определение 2.4. *Графом* называется пара (V, E), где

- V — множество вершин,
- $E \subseteq V^{(2)}$ — множество ребер.

Определение 2.5. *Реберно-взвешенным графом* называется граф $G = (V, E)$, для которого определена функция $w : E \to \mathbb{Q}$.

Невзвешенный граф будем считать частным случаем реберно-взвешенного, положив $w \equiv 1$.

В обоих случаях продолжим функцию w, положив $w(u, u) := 0$ для любой вершины u.

Иногда мы будем допускать в графе существование *кратных* ребер, граф с кратными ребрами называется *мультиграфом*.

В дальнейшем мы всегда будем рассматривать графы, у которых веса ребер неотрицательны, а для любых вершин u и v все ребра, их соединяющие, имеют одинаковый вес.

Определение 2.6. Определим также следующие понятия:

- *Путем* в графе $G = (V, E)$ будем называть последовательность вершин (с возможными повторениями), где любые две соседние вершины соединены ребром или совпадают. Путь называется *простым*, если в нем нет повторяющихся вершин.

- *Циклом* называется путь, у которого первая и последняя вершина совпадают. *Простой цикл* — цикл длины хотя бы три, в котором нет повторяющихся вершин (за исключением двух крайних).

- $P' = (u_1, \ldots, u_k)$ называется *подпутем* пути $P = (v_1, \ldots, v_m)$, если существует такое $i \in \{1, \ldots, m - k + 1\}$, что имеет место равенство $(u_1, \ldots, u_k) = (v_i, \ldots, v_{i+k-1})$. Если подпуть сам по себе является циклом, то его можно называть *подциклом*.

- $P' \subseteq P$ является обозначением того факта, что P' — подпуть P.

- Пусть $P_1 = (v_1, \ldots, v_r)$, $P_2 = (u_1, \ldots, u_q)$ причем вершины v_r и u_1 соединены ребром или совпадают. Тогда для путей P_1, P_2 определим конкатенацию следующим образом:

$$P = P_1 P_2 \iff P = (v_1, \ldots, v_r, u_1, \ldots, u_1).$$

  Заметим, что P — корректно определенный путь.

- Для любого пути $P = (v_1, \ldots, v_k)$ определим операцию обращения, результатом которой является путь $P^R = (v_k, \ldots, v_1)$.

Весом пути будем считать суммарный вес всех ребер, через которые этот путь проходит, с учетом кратности прохождения. Более формально, доопределим функцию w следующим образом: для произвольного пути $P = (v_1, v_2, \ldots, v_{m+1})$ положим $w(P) := \sum\limits_{j=1}^{m} w(v_j, v_{j+1})$.

Под длиной пути будем подразумевать число переходов, содержащихся в нем. Иначе говоря, длина пути $P = (v_1, v_2, \ldots, v_{m+1})$ равняется $\sum\limits_{j=1}^{m} \mathbf{1}_{v_j \neq v_{j+1}}$ и обозначается $|P|$.

Определение 2.7. Вершина u называется *достижимой* из вершины v, если существует путь, содержащий обе эти вершины.

Нетрудно проверить, что отношение достижимости является отношением эквивалентности, следовательно, разбивает множество вершин графа на классы эквивалентности, которые называются *компонентами связности*.

Определение 2.8. Граф называется *связным*, если он состоит из одной компоненты связности.

Определение 2.9. *Эйлеровым циклом* называется цикл, проходящий через все ребра графа ровно по одному разу. Граф называется *эйлеровым*, если в нем существует эйлеров цикл.

Введем обозначения:

- $N(v) = \{u \in V \mid \{u, v\} \in E\}$ — множество соседей вершины v,
- $deg(v) = |N(v)|$ — степень вершины v.

В частности, если $deg(v) = 0$, то такая вершина называется *изолированной*.

Широко известен следующий критерий эйлеровости графа:

Теорема 2.1. Для графа с кратными ребрами следующие утверждения равносильны:

- граф является эйлеровым и не содержит изолированных вершин,
- граф связен и степень любой его вершины четна.

В дальнейшем нам понадобится задача о существовании гамильтонова цикла в графе, поэтому введем соответствующие определения.

Определение 2.10. *Гамильтоновым циклом* называется цикл, проходящий через все вершины графа, причем не более чем по одному разу. Граф называется *гамильтоновым*, если в нем существует гамильтонов цикл.

Полиномиальных алгоритмов, отвечающих на вопрос о гамильтоновости произвольного графа, не известно. Более того, справедливо следующее утверждение:

Теорема 2.2 ([2])**.** Задача о гамильтоновости графа является NP-полной.

Определение 2.11. *Задача коммивояжера* формулируется следующим образом: заданы n городов и матрица $M \in \mathbb{Q}^{n \times n}$ расстояний между ними. Необходимо обойти все города (каждый ровно по одному разу) так, чтобы суммарное пройденное расстояние было минимальным.

В зависимости от матрицы M задача бывает *симметричной*, *ассиметричной*, *метрической* (если матрица M задает метрику).

Эта задача равносильна поиску гамильтонового цикла минимального веса в полном реберно-взвешенном графе.

Мы будем рассматривать постановку, в которой граф произвольный.

3 Историческая справка, обзор литературы

Вопрос о равенстве классов **P** и **NP** по сей день остается открытым и является одной из семи задач тысячелетия, за решение которой назначена премия в один миллион долларов. Впервые вопрос о равенстве этих классов был поставлен в 1971 году Стивеном Куком [3]. Результаты, подводившие к данной проблеме, также были получены в СССР Леонидом Анатольевичем Левиным [4]. В 1972 году Ричард Карп, используя полученные Стивеном Куком результаты, доказал NP-полноту большого числа задач [2], в частности, задачи о существовании гамильтонова цикла. Отсюда следует и NP-полнота задачи о коммивояжере.

Найти полиномиальный алгоритм для решения хотя бы какой-нибудь из NP-трудных задач до сих пор не удалось никому, тем не менее, ввиду их важности, активно разрабатываются алгоритмы, находящие приближенные решения за приемлимое время, как с гарантией результативности, так и без. В то время как для одних задач удается найти полиномиальные алгоритмы, гарантирующие результат, близкий к оптимальному (поиск вершинного покрытия [5], метрическая задача коммивояжера [6]), для других доказывается, что таких алгоритмов не существует, покуда $\mathbf{P} \neq \mathbf{NP}$ (таковы, например, задача о хроматическом числе графа [7] и общая задача коммивояжера [8]).

Первой известной задачей, связанной с путями на графах, можно считать задачу о существовании эйлерова цикла, а именно ее частный случай, известный как задача о семи кёнигсбергских мостах. Вопрос состоял в том, можно ли пройти по всем мостам таким образом, чтобы пересечь каждый не более чем единожды. 26 августа 1735 года Леонард Эйлер в ходе своей лекции привел доказательство невозможности существования такого пути.

В 1832 году в книге «Коммивояжёр — каким он должен быть и что он должен делать для привлечения заказов и уверенности в счастливой успешности своих сделок — от одного старого коммивояжера» была затронута проблема передвижения коммивояжера. Первая научная публикация, использующая термин “задача коммивояжера” (travelling salesman problem), датируется 1949 годом [9]. В 1974 году вышла статья, в которой доказывалась теоретическая оценка на отношение полученного решения к оптимальному для некотого алгоритма [6]. В дальнейшем, согласно [10], в работах [11] и [12] независимо было установлено, что для метрической задачи коммивояжера возможно за полиномиальное время отыскать решение, уступающее оптимальному не более чем в 1.5 раза. Этот результат долгое время оставался лучшим. Незначительное продвижение совсем недавно удалось осуществить группе ученых из Вашингтонского университета, предложив новый рандомизированный алгоритм, такой что ожидаемое решение уступает оптимальному менее чем в 1.5 раза [13].

Среди остальных алгоритмов стоит выделить те, идеи которых были заимствованы из биологии. К ним относятся муравьиный алгоритм [14], алгоритм пчелиной колонии [15], генетические алгоритмы [16]. Если первые два являются алгоритмами поиска пути на графе, то последние широко применяются в самых различных областях.

Особняком стоят точные алгоритм, такие как метод ветвей и границ, впервые описанный в [17], а также метод отсекающей плоскости [18], который в совокупности с первым позволил находить точные решения для некоторых примеров с числом городов, доходящим до десятков тысяч. Для многих примеров с миллионами городов найдены решения, для которых доказано, что они уступают оптимальному не более

чем на 2-3% [19].

Вне зависимости от того, требуется точное решение или приближенное, для ускорения последующих вычислений всегда хорошо найти эквивалентную задачу с как можно меньшей длиной записи, так как от нее зависит трудоемкость алгоритма [20].

4 Постановка задачи

4.1 Графическая задача коммивояжера

Здесь и далее мы зафиксируем некоторый способ бинарной записи пар вида (G, l), где $G = (V, E)$ — реберно-взвешенный граф, l — рациональное число. Более детально представление входных данных будет рассмотрено в разделе 4.3.

Определим язык **GTSP**, соответствующий графической задаче коммивояжера: если G — некоторый граф, а l — рациональное число, то пара (G, l) лежит в языке **GTSP** тогда и только тогда, когда существует цикл веса не более l, проходящий через все вершины графа G.

Мы будем часто использовать термин “путь коммивояжера” (иногда “цикл коммивояжера”), подразумевая при этом цикл, проходящий через все вершины графа. Обозначим $opt(G)$ — минимально возможный вес такого цикла в графе G.

Каждому пути коммивояжера сопоставим *граф пути* (вообще говоря, мультиграф), который получается из исходного графа приписыванием ребрам кратности, а именно, кратность каждого ребра полагается равной количеству раз, которое данное ребро встречается в пути. По определению будем считать, что ребро кратности 0 есть отсуствующее.

Граф, соответствующий некоторому оптимальному пути, будем называть *графом оптимального пути*.

Граф пути является эйлеровым графом без изолированных веришн, потому что путь коммивояжера, по которому граф был построен, и есть его эйлеров цикл, проходящий через все вершины.

Обратно, любой эйлеров граф без изолированных вершин порождается некото-

рым путем. Таким образом, мы получаем общеизвестный факт о том, что поиск оптимального цикла равносилен поиску эйлерова графа без изолированных вершин с минимальным суммарным весом ребер, который в качестве ребер содержит только ребра исходного графа, возможно большей кратности.

Из этого следует, что когда мы преобразовываем путь коммивояжера, нам достаточно следить лишь за тем, чтобы граф пути оставался связным, а степень всех вершин — четной.

В данной работе мы будем считать, что граф, подаваемый на вход, связен. Известно, что связность графа проверяется за линейное по числу вершин время (например, обходом в глубину), а если G — несвязный граф, то слово (G, l) заведомо не лежит в языке **GTSP** ни для какого l, поэтому ограничение на связные графы обосновано.

4.2 Редукция данных

Пусть заданы конечный алфавит Σ^* и язык $L \subseteq \Sigma^*$. Задача редукции данных заключается в том, чтобы сопоставить слову W из Σ^* другое слово W', имеющее как можно более короткую запись, так, чтобы $W \in L$ тогда и только тогда, когда $W' \in L$.

Определение 4.1 ([20])**.** Пусть $L \subseteq \Sigma^* \times \mathbb{N}$ — параметризованный язык. Тогда *кернелизацией* назовем алгоритм, отображающий $(x, k) \in \Sigma^* \times \mathbb{N}$ в $(x', k') \in \Sigma^* \times \mathbb{N}$ за не более чем полиномиальное время $poly(|x| + k)$ так, что:

1. $(x, k) \in L$ тогда и только тогда, когда $(x', k') \in L$,
2. существует вычислимая функция $f : \mathbb{N} \to \mathbb{N}$, такая что $|x'| + k' \leqslant f(k)$.

Слово (x', k') будем называть *ядром* (относительно параметра k), а f — *размером ядра*.

В дальнейшем нас будут интересовать исключительно ядра, имеющие полиномиальный размер.

Утверждение 4.1. Следующее равносильно:

- $(G, l) \in$ **GTSP** $\iff (G', l') \in$ **GTSP**,
- $opt(G) - opt(G') = l - l'$.

Доказательство. Заметим, слово (G, l) лежит в языке **GTSP** тогда и только тогда, когда $opt(G) \leqslant l$. Тогда первое условие перепишется так:

- $opt(G) \leqslant l \iff opt(G') \leqslant l'$.

Теперь равносильность очевидна. □

Пусть есть алгоритм, сопоставляющий слову (G, l) некоторое другое слово (G', l'). Благодаря утверждению 4.1, для доказательства его корректности, то есть того, что $(G, l) \in$ **GTSP** тогда и только тогда, когда $(G', l') \in$ **GTSP**, достаточно показать, что $opt(G) - opt(G') = l - l'$. Этим фактом мы будем активно пользоваться в дальнейшем.

4.3 Представление графа

В этом разделе обсудим представление входных данных. Будем отталкиваться от того, что все вершины и ребра графа можно представить бинарной строкой полиномиальной длины относительно количества вершин (например, такую запись может обеспечить задание графа матрицей смежности). Поскольку помимо этого нам нужно записывать веса ребер и рациональное число l, может получиться так, что суммарная длина входа окажется гораздо больше размеров самого графа. В данном разделе мы покажем, что всегда можно преобразовать веса ребер и параметр l так, чтобы длина их записи была полиномиальной по числу вершин. Таким образом, будет установлено, что для доказательства существования ядра, размер которого полиномиален по некоторому k, достаточно доказать, что число вершин в графе, полученном после кернелизации, полиномиально по k, а новый параметр k' вырос не более чем в полиномиальное число раз.

Здесь и далее $\langle x, y \rangle$ обозначает скалярное произведение векторов x и y.

Теорема 4.1 ([21])**.** Существует полиномиальный алгоритм, который по вектору $w \in \mathbb{Q}^r$ и натуральному N находит вектор $w' \in \mathbb{Z}^r$ такой что $\|w'\|_\infty \leqslant 2^{4r^3} N^{r(r+2)}$ и $sign(\langle w, b\rangle) = sign(\langle w', b\rangle)$ для любого вектора $b \in \mathbb{Z}^r$, $\|b\|_1 \leqslant N-1$.

Доказательство следующей леммы полностью аналогично [22].

Лемма 4.1. Для любой константы $c \in \mathbb{N}$ существует полиномиальный алгоритм, который по вектору $w \in \mathbb{Q}^r$ и числу $k \in \mathbb{Q}$ находит вектор $w' \in \mathbb{Z}^r$ и число $k' \in \mathbb{Z}$, такие что $\langle w', x\rangle \leqslant k' \iff \langle w, x\rangle \leqslant k$ для любого $x \in \{0, 1, \ldots, c\}^r$, причем вектор (w', k') может быть закодирован с помощью $O(r^4)$ бит. Более того, если вектор w не содержит отрицательных элементов, то и все элементы вектора w' тоже неотрицательны.

Доказательство. Применим теорему к вектору $y = (w_1, ..., w_r, -k) \in \mathbb{Q}^{r+1}$ и $N := c(r+1)+1$. Для любого $x \in \{0, 1, \ldots, c\}^{r+1}$ выполено

$$\|x\|_1 \leqslant c(r+1) = N-1,$$

поэтому для полученного $y' = (w'_1, ..., w'_r, -k') \in \mathbb{Z}^r$ имеет место $sign(\langle y, x\rangle) = sign(\langle y', x'\rangle)$, откуда следует, что $\langle w', x\rangle \leqslant k'$ тогда и только тогда, когда $\langle w, x\rangle \leqslant k$.

Для доказательства длины записи заметим

$$\|y'\|_\infty \leqslant 2^{4(r+1)^3}(cr+c+1)^{(r+1)(r+3)}.$$

Значит, каждый элемент вектора y' может быть записан с использованием не более чем

$$\log_2(2^{4(r+1)^3}(cr+c+1)^{(r+1)(r+3)}) = O(r^3)$$

бит. Поскольку размерность вектора $r+1$, суммарно получаем $O(r^4)$ бит.

Наконец, если $w'_i < 0$, то рассмотрим вектор x, у которого на i-й позиции стоит единица, а на остальных нули. $w'_i = \langle w', x\rangle \leqslant w'_i$, поэтому $w_i = \langle w, x\rangle \leqslant w'_i < 0$. □

В силу леммы 4.1 нам хотелось бы представить каждый путь коммивояжера вектором из $\{0, 1, \ldots, c\}^r$. Покажем, что достаточно взять $r = |E|$, $c = 2$.

Лемма 4.2. Существует оптимальный цикл коммивояжера, в котором любое ребро проходится не более одного раза в каждом из направлений.

Доказательство. Рассмотрим любой оптимальный путь P с минимально возможным числом ребер в соответствующем ему графе пути (с учетом кратностей). Покажем, что он и есть искомый.

Предположим, что коммивояжер посетил ребро $\{u, v\}$ более двух раз. Это значит, что в графе пути как минимум 3 ребра соединяют данные вершины. Удалим любые два из них. Связность не нарушится, четность степеней вершин не изменится, суммарный вес графа пути не увеличится. Поскольку число ребер в графе пути уменьшилось, мы получили противоречие с выбором пути P.

Докажем теперь, что коммивояжер посещает любое ребро не более чем по одному разу в каждом из направлений. Пусть не так. Без ограничения общности считаем, что коммивояжер два раза прошелся в направлении от u к v. Но тогда после первого прохода коммивояжер вернулся из вершины v в веришну u, минуя ребро $\{u, v\}$, следовательно, удаление ребер между этими вершинами не нарушит ни связности графа пути, ни четности степеней его вершин. Полученный таким образом граф будет задавать путь не большего веса, следовательно является графом оптимального пути со строго меньшим числом ребер, что противоречит выбору P. □

Пронумеруем все ребра в графе так, что $E = \{e_1, \cdots, e_m\}$. Обозначим вес ребра e_i как w_i. Тогда для любого пути P:

$$w(P) = \sum_{i=1}^{m} w_i x_i$$

где x_i — число раз, которое ребро e_i встречается в пути P. По лемме 4.2 для некоторого оптимального пути $x_i \in \{0, 1, 2\}$, $i = 1, \cdots, m$. Следовательно, мы можем ограничиться рассмотрением только таких путей.

Пусть дано слово (G, l). Построим вектор $\vec{w} = (w_1, \cdots, w_m)$ и применим к $(\vec{w}, l)$ лемму 4.1, получив $(\vec{w}', l')$. Теперь наше слово записывается длиной, полиномиальной по числу вершин в графе (так как $m < |V|^2$), и при этом выполнено следующее:

$opt(G) \leqslant l$ тогда и только тогда, когда в графе G с новым вектором весов $\vec{w}'$ существует цикл коммивояжера длины не превосходящей l', в котором любое ребро проходится не более чем дважды.

Более того, лемма 4.1 также утверждает, что после преобразования веса всех ребер остались неотрицательны. Поэтому в графе G с новым ветором весов $\vec{w}'$ существует оптимальный цикл, который также кодируется вектором из $\{0, 1, 2\}^m$. Отсюда следует:

Утверждение 4.2. $opt(G) \leqslant l$ тогда и только тогда, когда в графе G с новым вектором весов $\vec{w}'$ существует оптимальный цикл коммивояжера, вес которого не превосходит l'.

Утверждение 4.2 позволяет нам в дальнейшем считать длину записи любого слова (G, l) полиномиальной по числу вершин в графе.

5 Ядро относительно размера минимального разрезающего циклы набора ребер

В данном разделе будет рассмотрена кернелизация, возвращающая ядро полиномиального размера относительно мощности минимального разрезающего циклы набора ребер.

5.1 Кернелизация

Определение 5.1. Связный граф $G = (V, E)$ называется *деревом*, если он связен и не содержит простых циклов.

Определение 5.2. Граф $G = (V, E)$ называется *лесом*, если он состоит из одной или нескольких компонент связности, каждая из которых является деревом.

Хорошо известно следующее:

Теорема 5.1. Граф $G = (V, E)$ является деревом тогда и только тогда, когда он связен и $|E| = |V| - 1$.

Определение 5.3. Множество ребер $F \subseteq E$ называется *разрезающим циклы набором ребер* (по-английски *Feedback edge set*), если после его удаления получившийся граф является лесом.

Параметр **fes** равняется минимально возможной мощности такого множества.

Множество всех ребер E заведомо удовлетворяет данному условию, поэтому определение корректно.

Определение 5.4. Вершина $v \in V$ называется *листом*, если $deg(v) = 1$.

Определение 5.5. Простой путь $(v_0, \cdots, v_{m+1})$ назовем *цепью*, если $deg(v_1) = \ldots = deg(v_m) = 2$.

Пусть дано слово (G, l), которое нам надо проверить на принадлежность языку **GTSP**. Введем следующие правила редукции:

1. Если вершина u, инцидентная ребру $\{u, v\}$, является листом, то удалим его и уменьшим l на $2w(u, v)$.

2. Для любой цепи $(v_0, ..., v_{m+1})$ выберем ребро $\{v_j, v_{j+1}\}$ с максимальным весом (если таких несколько, то любое из них). После преобразуем граф следующим образом (рис. 5.1 и 5.2):

   - удалим вершины $\{v_1, \cdots, v_m\} \backslash \{v_j, v_{j+1}\}$,
   - добавим ребро $\{v_0, v_j\}$, положим $w(v_0, v_j) := \sum\limits_{i=1}^{j-1} w(v_i, v_{i+1})$,
   - добавим ребро $\{v_{j+1}, v_{m+1}\}$, положим $w(v_{j+1}, v_{m+1}) := \sum\limits_{i=j+1}^{m} w(v_i, v_{i+1})$.

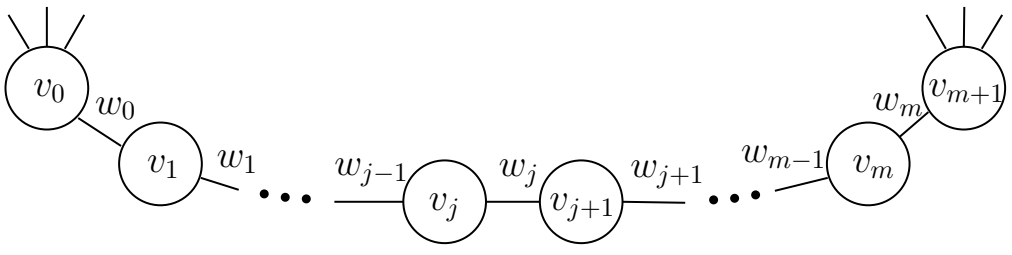

Рис. 5.1: До применения правила 2

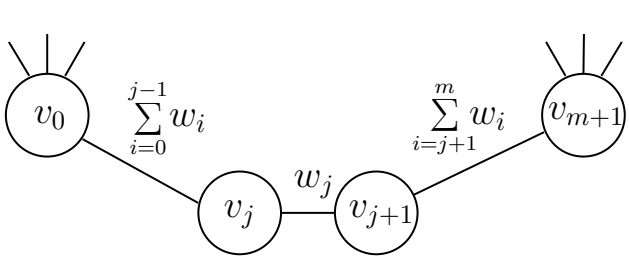

Рис. 5.2: После применения правила 2

Утверждение 5.1. Выполнение каждого из этих правил полиномиально по времени от длины входа.

Доказательство. Следует из того, что степени всех вершин определяются за время, полиномиальное по числу вершин в графе, а время выполнения арифметических операций над рациональными числами также полиномиально по длине их записи. □

Определим кернелизацию, которая заключается в последовательном применении первого и второго правил до тех пор, пока граф не перестанет уменьшатся. Заметим, что если алгоритм отработал k итераций, то число оставшихся вершин не превосходит $|V| - k$, откуда следует, что общее число итераций не превосходит $|V|$, поэтому данная процедура полиномиальна по длине входа.

Для доказательства того факта, что описанный алгоритм действительно является кернелизацией для языка **GTSP**, параметризованного значением **fes**, остается показать, что, во-первых, он не влияет на принадлежность рассматриваемого слова языку, а во-вторых, что размер ядра действительно ограничен некоторой вычислимой функцией от параметра. Всё это будет показано далее в разделах 5.2 и 5.3.

5.2 Корректность

В этом разделе мы покажем, что получившееся в результате применения редукции слово (G', l') лежит в языке **GTSP** тогда и только тогда, когда в нем лежит исходное слово (G, l).

Начнем с доказательства корректности первого правила.

Утверждение 5.2. Пусть v — лист в реберно-взвешенном графе $G = (V, E)$, причем $\{u, v\} \in E$. Обозначим $G' = (V', E')$, где

- $V' := V \backslash \{v\}$,
- $E' := E \backslash \{\{u, v\}\}$.

Тогда $opt(G') = opt(G) - 2w(u, v)$.

Доказательство. Нассмотрим граф оптимального пути в G. Удалим из него вершину v. Связность, очевидно, нарушена не будет, четность степеней тоже, поскольку степень вершины u уменьшится ровно на степень вершины v, которая обязана быть четной в силу эйлеровости графа пути. Таким образом, $opt(G') \leqslant opt(G) - 2w(u, v)$.

Обратно, если к графу оптимального пути в G' добавить вершину v и соединить ее ребром кратности 2 с вершиной u, то мы получим корректный граф пути в G с суммарным весом $opt(G') + 2w(u, v)$, поэтому $opt(G) \leqslant opt(G') + 2w(u, v)$. □

Перейдем к доказательсву корректности второго правила.

Лемма 5.1. В графе пути все ребра между вершинами цепи имеют одинаковую четность.

Доказательство. Предположим противное. Рассмотрим вершину v_i, где i — минимальное среди таких индексов, что ребро $\{v_i, v_{i+1}\}$ имеет четность, отличную от четности ребра $\{v_0, v_1\}$. Из вершины v_i выходят два ребра, $\{v_{i-1}, v_i\}$ и $\{v_i, v_{i+1}\}$, разной четности, следовательно, суммарная степень вершины v_i нечетна, что противоречит эйлеровости графа пути. □

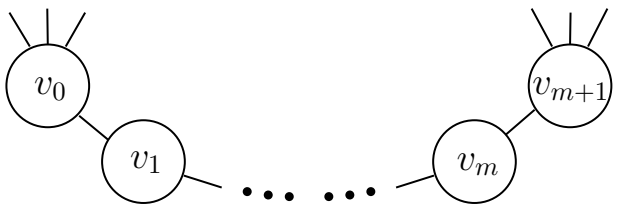

Рис. 5.3: все ребра цепи имеют кратность 1

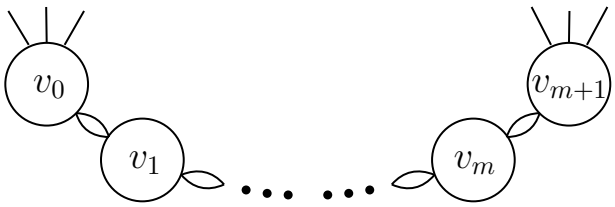

Рис. 5.4: все ребра цепи имеют кратность 2

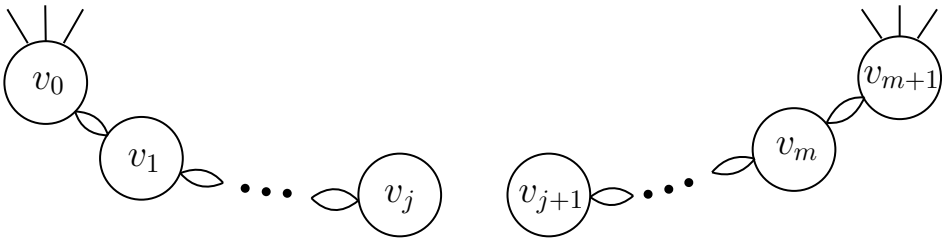

Рис. 5.5: одно ребро цепи имеет нулевую кратность, все остальные кратности 2

Лемма 5.2. Существует граф оптимального пути, в котором любая цепь имеет один из трех видов, указанных на рисунках 5.3, 5.4, 5.5.

Доказательство. По лемме 4.2 можем рассмотреть граф оптимального пути, в котором крастность любого ребра не превосходит двух. Покажем, что он и есть искомый. Разберем возможные случаи с учетом леммы 5.1:

1. Все ребра нечетной кратности.

   Тогда все ребра ровно кратности 1, что соответвует рис. 5.3.

2. Все ребра четной кратности.

   Тут возможны еще два варианта:

   (a) Все ребра кратности 2.

   Этот вариант соответствует рис. 5.4.

   (b) Есть ребро кратности 0 (отсутствующее).

   Тогда из связности графа пути следует, что такое ребро может быть ровно одно, что соответствует рис. 5.5.

□

Лемма 5.3. Если в графе оптимального пути имеет место случай, указанный на рис. 5.5, то $w(v_j, v_{j+1}) = \max\{w(v_i, v_{i+1}) \mid i \in \{0, \cdots, m\}\}$. Более того, в качестве пары вершин (v_j, v_{j+1}) могут выступать любые две вершины, ребро между которыми имеет максимальный вес в цепи.

Доказательство. Докажем от противного. Пусть $w(v_j, v_{j+1}) < w(v_k, v_{k+1})$. Удалим ребра между вершинами v_k и v_{k+1}, добавим два ребра $\{v_j, v_{j+1}\}$. Эйлеровость графа пути не нарушится, а суммарный вес ребер уменьшится, что противоречит оптимальности рассматриваемого пути.

Заметим, что такую же операцию мы можем выполнять, не нарушая оптимальности пути, в том случае, если $w(v_j, v_{j+1}) = w(v_k, v_{k+1})$. □

Лемма 5.4. Пусть $(v_0, \ldots, v_{m+1})$ — цепь в графе G, и после применения к ней второго правила получился граф G'. Тогда $opt(G) = opt(G')$.

Доказательство. Легко проверяется разбором всех трех случаев. □

Таким образом, корректность второго правила также доказана.

5.3 Размер ядра

В этом разделе мы покажем, что число вершин в получившимся графе $G' = (V', E')$ полиномиально по параметру $\mathbf{fes}(G)$. Заметим, что если $V' = \varnothing$, то утверждение очевидно, поэтому будем рассматривать только непустые графы.

Лемма 5.5. Пусть $G = (V, E)$ — связный граф, а граф $T = (V, E')$ получен удалением разрезающего циклы набора ребер F минимального размера. Тогда T — дерево.

Доказательство. Предположим противное. Рассмотрим любое ребро e в исходном графе G, соединяющее разные компоненты связности в графе T. Заметим, что после добавления ребра e граф T останется лесом. Значит, $F' := F \backslash \{e\}$ тоже является

разрезающим циклы набором ребер, причем строго меньшей мощности, что противоречит условию. □

Лемма 5.6. Для связного графа $G = (V, E)$ параметр **fes**(G) равен $|E| - |V| + 1$.

Доказательство. Пусть F – разрезующий циклы набор ребер, $|F| =$ **fes**(G), и после его удаления остается граф $T = (V, E')$. По лемме 5.5 граф T — дерево, поэтому $|E'| = |V| - 1$ (теорема 5.1). Осталось заметить, что $|E| - |F| = |E'|$. □

Лемма 5.7. Обозначим $G = (V, E)$ — исходный граф, $G' = (V', E')$ — полученный после применения алгоритма. Тогда **fes**$(G) =$ **fes**(G')

Доказательство. По лемме 5.6 достаточно проверить, что ни одна из операций не изменяет $|E| - |V|$. □

Теорема 5.2. Размер ядра есть некоторый полином по параметру **fes**.

Доказательство. Докажем, что если в связном графе $G = (V, E)$ нет листьев и длина всех цепей ограничена сверху константой m (в нашем случае $m = 3$), то $|V| = O($**fes**$)$. По лемме 5.7 и рассуждениям, приведенным в разделе 4.3, этого будет достаточно.

Посчитаем, сколько вершин входят в минимальный разрезающий циклы набор ребер F. Рассмотрим дерево T, получившееся после удаления F. Разобьем множество его вершин на два подмножества:

- V_1 – внутренние вершины цепей, т.е. имеющие степень ровно два,
- V_2 – концевые вершины цепей, т.е. имеющие степень, отличную от двух.

Множество V_1, в свою очередь, разобьем на еще на два подмножества:

- V_1' – вершины, инцидентные ребрам из F,
- V_1'' – остальные.

Заметим, что $|V_1'|(m-1) \geqslant |V_1''|$, откуда $|V_1'| \geqslant \frac{1}{m}|V_1|$. Избавимся теперь от вершин V_1, "стянув" цепи. Число листьев хотя бы $\frac{1}{2}|V_2|$, каждый лист обязан входить в ребра

из F. Суммарное количество вершин, входящих в ребра из F, оказывается равным хотя бы

$$\frac{1}{m}|V_1| + \frac{1}{2}|V_2| \geqslant \frac{1}{m}|V|.$$

Отсюда получаем, что $2|F| \geqslant \frac{1}{m}|V|$, т.е. $|V| \leqslant 2m \cdot \textbf{fes} = O(\textbf{fes})$. □

6 Ядро относительно размера минимального вершинного покрытия

В данном разделе будет рассмотрена кернелизация, возвращающая ядро полиномиального размера относительно мощности минимального вершинного покрытия.

6.1 Кернелизация

Определение 6.1. Множество $C \subseteq V$ будем называть *вершинным покрытием* (по-английски *vertex cover*), если для любого ребра $\{u, v\} \in E$ выполнено $\{u, v\} \cap C \neq \varnothing$.

Размер минимального вершинного покрытия будем обозначать **vc**. Поиск такого множества является, вообще говоря, NP-трудной задачей. Тем не менее, существуют полиномиальные алгоритмы, позволяющие искать приближенное решение с доказуемой точностью. Таков, например, следующий алгоритм, представленный в [5]:

Algorithm 1: Алгоритм поиска независимого подмножества

$G = (V, E)$;
$I := \varnothing$;
while $E \neq \varnothing$ **do**
  $\{u, v\} \in E$;
  $I := I \cup \{u, v\}$;
  $E := E \backslash \{e \in E \mid e \cap \{u, v\} \neq \varnothing\}$;
end
return I;

Теорема 6.1 ([5])**.** Пусть I – результат работы алгоритма. Тогда $|I| \leqslant 2\mathbf{vc}$.

Для удобства будем использовать следующие обозначения:

- $N(u,v) := N(u) \cap N(v)$,
- $w(s,v,u) := w(s,v) + w(s,u)$,
- $w_{min}(s) := \min\limits_{v \in N(s)} w(s,v)$.

Для произвольного реберно-взвешенного графа G с вершинным покрытием C определим семейство множеств

$$\mathcal{S} := \{S_{\{u,v\}} \mid \{u,v\} \in C^{(2)}\},$$

такое что

1. $S_{\{u,v\}} \subseteq N(u,v) \backslash C$,
2. $S_{\{u,v\}} \cap S_{\{u',v'\}} = \varnothing$,
3. $|S_{\{u,v\}}| \leqslant 2$ для любого $\{u,v\} \in C^{(2)}$,
4. $|\bigcup \mathcal{S}| \to max$.

В дальнейшем для удобства будем обозначать $S := \bigcup \mathcal{S}$.

Свойство семейства удволетворять данным условиям будем называть *периферийностью*, само семейство $\mathcal{S}$ — *периферийным*. Если же выполнены все условия, кроме последнего, то такое семейство назовем *псевдопериферийным*. Легко проверяется справедливость следующего утверждения:

Лемма 6.1. Приведенные ниже операции над семейством $\mathcal{S}$ не нарушают его псевдопериферийности:

1. добавление вершины из $N(u,v) \backslash (C \cup S)$ в множество $S_{\{u,v\}}$ в случае, если $|S_{\{u,v\}}| < 2$
2. удаление любой вершины из $S_{\{u,v\}}$

3. перенос вершины $t \in S_{\{u,v\}} \cap N(u', v')$ из множества $S_{\{u,v\}}$ в множество $S_{\{u',v'\}}$ в том случае, если $|S_{\{u',v'\}}| < 2$.

Более того, первая операция увеличивает мощность множества S на 1, вторая операция ее уменьшает на 1, а третья ее никак не меняет, т.е. сохраняет периферийность.

Введем функцию $\varphi : V^3 \to \mathbb{Q}$:

$$\varphi(s, u, v) = w(s, u) + w(s, v) - 2w_{min}(s) = w(s, u, v) - 2w_{min}(s).$$

Продолжим функцию φ для семейства $\mathcal{S}$:

$$\varphi(\mathcal{S}) = \sum_{S_{\{u,v\}} \in \mathcal{S}} \sum_{s \in S_{\{u,v\}}} \varphi(s, u, v).$$

Если на периферийном семействе $\mathcal{S}$ достигается минимум функции φ, то будем будем называть такое семейство минимальным. Иначе говоря, мы упорядочили периферийные семейства по значению φ.

В разделе 6.2 будет показано, что для любого минимального периферийного семейства $\mathcal{S}$ удаление произвольной вершины v, принадлежащей $V \backslash (C \cup S)$, влечет за собой уменьшение оптимального пути ровно на $2w_{min}(v)$.

Пусть заданы реберно-взвешенный граф G и рациональное число l, и слово (G, l) необходимо проверить на принадлежность языку **GTSP**. Порядок действий будет следующим:

1. С помощью алгоритма 1 найти вершинное покрытие C, такое что $|C| \leqslant 2\mathbf{vc}$.

2. Найти любое минимальное периферийное семейство $\mathcal{S}$.

3. Удалить все вершины, лежащие в $R := V \backslash (C \cup S)$, уменьшить значение l на $2 \sum\limits_{v \in R} w_{min}(v)$.

Отдельно стоит рассмотреть случай, когда множество C мощности 1. В таком случае эта единственная вершина инцидента всем ребрам, т.е. граф имеет форму звезды. Все остальные вершины являются листьями, их мы умеем удалять (см. раздел 5).

В любом случае, алгоритм является полиномиальным по числу вершин в графе, если предположить, что мы умеем искать минимальное периферийное семейство за полиномиальное время (рассматривается в разделе 6.3). Отметим также, что количество вершин в оставшемся графе равняется

$$|C| + |S| \leqslant |C| + |C|^2 \leqslant 2\mathbf{vc} + 4\mathbf{vc}^2 = O(\mathbf{vc}^2),$$

а параметр **vc** не превосходит изначального значения.

6.2 Корректность

Все рассуждения в этом разделе будут проводиться для произвольного фиксированного реберно-взвешенного графа $G = (V, E)$ с заданным вершинным покрытием C.

Рассмотрим P — какой-нибудь путь коммивояжера. Для любых различных вершин u и v из C определим $T^P_{(u,v)}$ следующим образом:

$$T^P_{(u,v)} := \{u' \in V \backslash C \mid (u, u', v) \subseteq P\}.$$

Положим

- $T^P_{\{u,v\}} := T^P_{(u,v)} \cup T^P_{(v,u)}$,
- $T^P := \bigcup\limits_{\{u,v\} \in C^{(2)}} T^P_{\{u,v\}}$.

Лемма 6.2. Существует оптимальный путь P, такой что $|T^P_{(u,v)}| \leqslant 1$ для любых вершин u и v из C.

Доказательство. Рассмотрим оптимальный путь P с минимально возможной мощностью множества T^P. Покажем, что он и есть искомый. Пусть нашлось такое $T^P_{(v_1,v_2)}$, что его мощность хотя бы 2. Рассмотрим t_1 и t_2, лежащие в $T^P_{(v_1,v_2)}$. Тогда P можно представить в следующем виде:

$$P = P_1(v_1, t_1, v_2)P_2(v_1, t_2, v_2)P_3.$$

Положим

$$P' := P_1(v_1)P_2^R(v_2)P_3.$$

Заметим, P' проходит через все вершины графа, за исключением, быть может, вершин t_1 и t_2. Введем обозначения:

- $\alpha_i := \arg\min\limits_{q\in\{1,2\}} w(v_q, t_i),\ i = 1, 2,$
- $\beta_i := \{1, 2\}\backslash\{\alpha_i\}.$

Преобразуем P': когда коммивояжер первый раз оказывается в вершине v_{α_i} ($i = 1, 2$), он делает переход $(v_{\alpha_i}, t_i, v_{\alpha_i})$, далее продолжает следовать по пути P'.

Полученный таким образом путь обозначим $\hat{P}$. Заметим, что $\hat{P}$, в отличие от P', проходит уже через все вершины графа, а потому является корректным циклом коммиворяжера. Более того,

$$w(\hat{P}) - w(P) = w(v_{\alpha_1}, t_1) + w(v_{\alpha_2}, t_2) - w(v_{\beta_1}, t_1) - w(v_{\beta_2}, t_2) \leqslant 0,$$

поэтому он еще и оптимален. Далее, $T^{\hat{P}} = T^P\backslash\{t_1, t_2\}$, поскольку обращение подциклов не меняет никакое из множеств $T_{\{u,v\}}$. Значит, $|T^{\hat{P}}| < |T^P|$, что противоречит выбору пути P. □

Лемма 6.3. Существует оптимальный путь P, такой что $|T^P_{\{u,v\}}| \leqslant 2$ для любых вершин u и v из C.

Доказательство. Рассмотрим оптимальный путь P, для которого выполнены условия леммы 6.2. Тогда $T^P_{\{u,v\}} = T^P_{(u,v)} \cup T^P_{(v,u)}$, поэтому $|T^P_{\{u,v\}}| \leqslant |T^P_{(u,v)}| + |T^P_{(v,u)}| \leqslant 2$. □

Введем функцию $\theta_P : C^2 \times C^2 \to \mathbb{N}$:

$$\theta_P((u_1, v_1), (u_2, v_2)) := \begin{cases} |S_{\{u_1,v_1\}} \cap T^P_{(u_2,v_2)}|, & \{u_1, v_1\} \neq \{u_2, v_2\} \\ 0, & \{u_1, v_1\} = \{u_2, v_2\} \end{cases}$$

Положим $\Theta_P := \sum\limits_{x,y\in C^{(2)}} \theta(x, y)$. Теперь определим следующее отношение:

$$(u_1, v_1) \mapsto (u_2, v_2) \iff \theta((u_1, v_1), (u_2, v_2)) > 0.$$

Опишем те рассуждения, которые привели нас данным обозначениям. Наша цель — показать, что существует оптимальный путь P, такой что $T^P \subseteq S$ для некоторого фиксированного периферийного семейства $\mathcal{S}$. Пусть это не выполнено, т.е. нашлись такая пара вершин $(u, v) \in C^2$ и такая вершина $t \in V \backslash (C \cup S)$, что $t \in T_{(u,v)}$. Тогда, как мы покажем, $S_{\{u,v\}} \neq \varnothing$, и для любой вершины $s \in S_{\{u,v\}}$ возможны два случая:

1. вершина s лежит в $V \backslash (C \cup T)$,
2. нашлась такая пара различных вершин $(u', v') \in C^2$, что $s \in T_{(u,v)}$.

Функция φ была подобрана ровно так, чтобы в первом случае путь P можно было перестроить необходимым образом, сохранив его оптимальность. Второй случай означает, что $|S_{\{u,v\}} \cap T^P_{(u',v)}| > 0$, поэтому в дальнейшем мы будем минимизировать Θ_P.

Лемма 6.4. Если P — такой оптимальный путь, что $|T^P_{(u,v)}| \leqslant 1$ для любых u, v из C, то Θ_P равняется количеству таких пар $((u, v), (u', v'))$, что $(u, v) \mapsto (u', v')$.

Доказательство. Заметим, $\theta_P((u_1, v_1), (u_2, v_2) \leqslant |T^P_{(u_2,v_2)}| \leqslant 1$. Поэтому множество значений функции θ_P имеет вид $\{0, 1\}$, причем $\theta_P((u, v), (u', v')) = 1$ в том и только том случае, когда $(u, v) \mapsto (u', v')$, откуда и следует требуемое. □

Лемма 6.5. Любой подпуть содержится в оптимальном пути коммивояжера не более одного раза.

Доказательство. Следует из того, что любое ребро посещается в каждом из направлений не более чем по одному разу (лемма 4.2). □

Лемма 6.6. Пусть зафиксировано минимальное периферийное семейство $\mathcal{S}$ и для некоторого пути коммивояжера P выполнено $(u_1, v_1) \mapsto \cdots \mapsto (u_m, v_m) \mapsto (u_1, v_1)$. Тогда существует путь P', для которого верно $w(P') \leqslant w(P)$ и $\Theta_{P'} < \Theta_P$.

Доказательство. Для удобства выкладок обозначим $u_{m+1} := u_1$, $v_{m+1} := v_1$, а также $T^P_i := T^P_{(u_i,v_i)}$, $S_i := S_{\{u_i,v_i\}}$ для $i = 1, \cdots, m + 1$. Рассмотрим $s_1 \in S_1 \cap T^P_2, \ldots, s_m \in S_m \cap T^P_{m+1}$. Перестроим путь коммивояжера следующим образом: для

$i = 1, \ldots, m$ каждый подпуть (u_{i+1}, s_i, v_{i+1}) заменим на $(u_{i+1}, s_{i+1}, v_{i+1})$. Очевидно, что получившийся путь P' проходит через все вершины, поэтому является корректным циклом коммивояжера. По лемме 6.5 любой подпуть встречается в оптимальном пути ровно единожды, поэтому суммарный вес пути изменится на величину $\sum\limits_{i=1}^{m} w(s_i, v_i, u_i) - \sum\limits_{i=1}^{m} w(s_i, v_{i+1}, u_{i+1})$.

Покажем, что $\sum\limits_{i=1}^{m} w(s_i, v_i, u_i) - \sum\limits_{i=1}^{m} w(s_i, v_{i+1}, u_{i+1}) \leqslant 0$, откуда будет следовать $w(P') \leqslant w(P)$. Для этого построим периферийное семейство $\mathcal{S}'$, а именно переместим каждое s_i в соответствующее множество S_{i+1} (лемма 6.1). Воспользуемся минимальностью $\mathcal{S}$:

$$\varphi(\mathcal{S}) \leqslant \varphi(\mathcal{S}')$$

$$\sum_{i=1}^{m} \varphi(s_i, v_i, u_i) \leqslant \sum_{i=1}^{m} \varphi(s_i, v_{i+1}, u_{i+1})$$

$$\sum_{i=1}^{m} w(s_i, v_i, u_i) - 2\sum_{i=1}^{m} w_{min}(s_i) \leqslant \sum_{i=1}^{m} w(s_i, v_{i+1}, u_{i+1}) - 2\sum_{i=1}^{m} w_{min}(s_i)$$

$$\sum_{i=1}^{m} w(s_i, v_i, u_i) \leqslant \sum_{i=1}^{m} w(s_i, v_{i+1}, u_{i+1})$$

Теперь покажем, что для любой пары $((u, v), (u', v'))$ выполняется вложение $S_{\{u,v\}} \cap T^{P'}_{(u',v')} \subseteq S_{\{u,v\}} \cap T^{P}_{(u',v')}$. Пусть не так, т.е. нашлась вершина s, которая лежит в $(S_{\{u,v\}} \cap T^{P'}_{(u',v')}) \backslash (S_{\{u,v\}} \cap T^{P}_{(u',v')})$. Очевидно, что $s = s_i \in \{s_1, \cdots, s_m\}$. Тогда $\{u, v\} = \{u_i, v_i\}$, потому что все множества семейства $\mathcal{S}$ попарно не пересекаются, а s_i, как мы знаем, лежит в $S_{\{u_i,v_i\}}$. С другой стороны, вершина s_i была перенесена в $T_{(u',v')}$, откуда получаем, что (u', v') также обязано совпадать с (u_i, v_i). В итоге имеем либо $(u_i, v_i) \mapsto (u_i, v_i)$, либо $(u_i, v_i) \mapsto (u_i, v_i)$, но оба варианта невозможны в силу того, как определялось отношение.

Для завершения доказательства осталось заметить, что для $i = 1, \ldots, m$ выполнено $s_i \in (S_{\{u_i,v_i\}} \cap T^{P}_{(u_{i+1},v_{i+1})}) \backslash (S_{\{u_i,v_i\}} \cap T^{P'}_{(u_{i+1},v_{i+1})})$, откуда и следует $\Theta_{P'} < \Theta_P$. □

Лемма 6.7. Пусть $\mathcal{S}$ — минимальное периферийное семейство, P — произвольный цикл коммивояжера, а вершина u содержится в множестве $N(u_0, v_0) \backslash (S \cup C)$. Тогда

1. $|S_{\{u_0,v_0\}}| = 2$

2. Для любых u_k, v_k, таких что

$$(u_0, v_0) \mapsto (u_1, v_1) \mapsto \cdots \mapsto (u_k, v_k)$$

выполнено $|S_{\{u_k,v_k\}}| = 2$.

Доказательство.

1. В противном случае u можно было бы добавить в $S_{\{u_0,v_0\}}$ (лемма 6.1), что противоречит максимальности множества S по мощности.

2. Обозначим $S_i := S_{\{u_i,v_i\}}$. Будем доказывать индукцией по k.

   Пусть выполнено $(u_0, v_0) \mapsto (u_1, v_1)$. Это значит, что существует вершина $s_0 \in S_0 \cap T_1$. Если $|S_1| < 2$, то по лемме 6.1 можем перенести вершину s_0 из S_0 в S_1, после чего добавить u в S_0. В результате этих действий мощность множества S строго возрастет, что противоречит периферийности исходного семейства $\mathcal{S}$. База доказана.

   Пусть теперь $|S_0| = \cdots = |S_{k-1}| = 2$. Рассмотрим последовательсть вершин $s_0 \in S_0 \cap T_{(v_1,u_1)}, \cdots, s_{k-1} \in S_{k-1} \cap T_{(v_k,u_k)}$. Если $|S_k| < 2$, то переместим s_{k-1} из S_{k-1} в S_k, s_{k-2} из S_{k-2} в S_{k-1} и так далее, вплоть до s_0 (лемма 6.1). Наконец, на освободившееся место в множестве S_0 переместим вершину u. Суммарная мощность S увеличилась, чего быть не может.

□

Лемма 6.8. Пусть зафиксировано любое минимальное периферийное семейство $\mathcal{S}$. Тогда существует оптимальный путь P, такой что выполнено вложение $T^P_{\{u,v\}} \subseteq S$ для любых u и v из C.

Доказательство. Рассмотрим любой такой оптимальный путь P, что $|T^P_{\{u,v\}}| \leqslant 2$ для любых вершин u и v из C (существует в силу леммы 6.3), и при этом Θ_P минимально. Покажем, что он и является искомым. Пусть не так, т.е. нашлась такая пара вершин (u_1, v_1), для которых существует вершина t, лежащая в множестве $T_{(u_1,v_1)} \backslash S$.

Поскольку циклов по отношению $\mapsto$ в силу минимальности Θ_P быть не может (лемма 6.6), то переходя по данному отношению, начиная из вершин (u_1, v_1), мы рано или поздно придем в сток:

$$(u_1, v_1) \mapsto \cdots \mapsto (u_{m-1}, v_{m-1}) \mapsto (u_m, v_m).$$

Обозначим $S_i := S_{\{u_i,v_i\}}$, $T_i^P := T^P_{(u_i,v_i)}$ для $i = 1, \cdots, m$. По лемме 6.7 $|S_1| = \cdots = |S_m| = 2$.

Пусть $s_i \in S_i \cap T^P_{i+1}$, $i = 1, \cdots, m-1$. Тогда $s_{m-1} \in S_{m-1} \cap T^P_m$, поэтому существует $s_m \in S_m \backslash T^P_m$ такое что $s_m \notin T^P$ (потому что (u_m, v_m) — сток).

Перестроим путь коммивояжера следующим образом: заменим переход (u_1, t, v_1) на (u_1, s_1, v_1), переход (u_2, s_1, v_2) заменим на (u_2, s_2, v_2) и так далее, вплоть до перехода (u_m, s_{m-1}, u_m), который мы заменим на (u_m, s_m, u_m). Поскольку вершина s_m не лежала в T^P, то ее посещение имело вид (x, s_m, x), причем $w(s_m, x) = w_{min}(s_m)$, так как мы рассматриваем оптимальный путь. Теперь же мы просто выкинем этот подцикл из нашего пути, а вместо него добавим (y, t, y), где y — любая такая вершина, что $w(y, t) = w_{min}(t)$.

Обозначим получившийся путь P'. Очевидно, что P' проходит через все вершины, а потому корректен. Покажем, что выполнено неравенство

$$\sum_{i=1}^{m} w(s_i, v_i, u_i) + 2w_{min}(t) \leqslant w(t, v_1, u_1) + \sum_{i=1}^{m-1} w(s_i, v_{i+1}, u_{i+1}) + 2w_{min}(s_m)$$

из которого будет следовать оптимальность P'. Переместим t в S_1, s_1 в S_2, s_2 в S_3 и так далее. Наконец, вершину s_m исключим из S_m. Получим новое периферийное семейство $\mathcal{S}'$ (лемма 6.1). Тогда

$$\varphi(S) \leqslant \varphi(S')$$

$$\sum_{i=1}^{m} \varphi(s_i, v_i, u_i) \leqslant \varphi(t, v_1, u_1) + \sum_{i=1}^{m-1} \varphi(s_i, v_{i+1}, u_{i+1})$$

$$\sum_{i=1}^{m} w(s_i, v_i, u_i) - 2\sum_{i=1}^{m} w_{min}(s_i) \leqslant w(t, v_1, u_1) + \sum_{i=1}^{m-1} w(s_i, v_{i+1}, u_{i+1}) - \\ -2w_{min}(t) - 2\sum_{i=1}^{m-1} w_{min}(s_i)$$

$$\sum_{i=1}^{m} w(s_i, v_i, u_i) + 2w_{min}(t) \leqslant w(t, v_1, u_1) + \sum_{i=1}^{m-1} w(s_i, v_{i+1}, u_{i+1}) + 2w_{min}(s_m).$$

Покажем, что для любой пары $((u, v), (u', v'))$ выполнено $S_{\{u,v\}} \cap T^{P'}_{(u',v')} \subseteq S_{\{u,v\}} \cap T^{P}_{(u',v')}$. Пусть не так, т.е. нашлась вершина s, которая лежит в $(S_{\{u,v\}} \cap T^{P'}_{(u',v')}) \backslash (S_{\{u,v\}} \cap T^{P}_{(u',v')})$. Очевидно, что $s = s_i \in \{s_1, \cdots, s_m\}$. Тогда $\{u, v\} = \{u_i, v_i\}$, потому что все множества семейства $\mathcal{S}$ попарно не пересекаются, а s_i, как мы знаем, лежит в $S_{\{u_i,v_i\}}$. С другой стороны, вершина s_i была перенесена в $T_{(u',v')}$, откуда получаем, что (u', v') также обязано совпадать с (u_i, v_i). В итоге имеем либо $(u_i, v_i) \mapsto (u_i, v_i)$, либо $(u_i, v_i) \mapsto (u_i, v_i)$, но оба варианта невозможны в силу того, как определялось отношение.

Для завершения доказательства осталось заметить, что для $i = 1, \ldots, m$ выполнено $s_i \in (S_{\{u_i,v_i\}} \cap T^{P}_{(u_{i+1},v_{i+1})}) \backslash (S_{\{u_i,v_i\}} \cap T^{P'}_{(u_{i+1},v_{i+1})})$, откуда и следует $\Theta_{P'} < \Theta_P$, что противоречит выбору пути P. □

Теорема 6.2. Удаление любой вершины $v \in V \backslash (C \cup S)$ ведет к уменьшению веса оптимального цикла ровно на $2w_{min}(v)$.

Доказательство. Прежде всего заметим, что никакие две вершины из $V \backslash C$ не связаны друг с другом ребрами. Действительно, для любого ребра, выходящего из $t \in V \backslash C$ второй его конец обязан принадлежать множеству C по определению вершинного покрытия.

Поскольку вершины из $V \backslash C$ не связаны друг с другом ребрами, то любое посещение $t \in V \backslash C$ имеет один из двух возможных видов: либо (u, t, v), либо (u, t, u), где $u, v \in C$. По лемме 6.8 найдется оптимальный путь, такой что любое посещение вершины $t \in V \backslash (C \cap S)$ имеет вид (u, t, u), где $u \in C$, причем из оптимальности пути также следует, что $w(u, t) = w_{min}(t)$, и что вершина t посещается таким образом ровно один раз.

Выкинем подпуть (u, t, u). Мы получили новый путь, вес которого меньше ровно на $2w_{min}(t)$ и который содержит все вершины, кроме вершины t.

Обратно, для любого пути, содержащего все вершины, кроме вершины t, можно добавить посещение вида (u, t, u), где $w(u, t) = w_{min}(t)$. □

6.3 Поиск минимального периферийного семейства

Рассмотрим способ построения минимального периферийного семейства $\mathcal{S}$ за полиномиальное по размеру графа время. *Сетью* мы будем называть граф, для которого заданы:

- две вершины: сток и исток,
- $\alpha : E \to \mathbb{Q}_{\geqslant 0}$ — пропускная способность ребра,
- $c : E \to \mathbb{Q}_{\geqslant 0}$ — стоимость ребра.

Рассмотрим двудольный граф, где в качестве первой доли D_1 берется множество $C^{(2)}$, а в качестве второй доли D_2 — множество $V \backslash C$, причем $\{u, u'\} \in C^{(2)}$ соединим ребром с $v \in V \backslash C$ тогда и только тогда, когда $v \in N(u, u')$. Добавим также исток s и сток t, после чего s соединим со всеми вершинами из D_1, а t — со всеми из D_2. Для того, чтобы определить сеть, остается назначить каждому ребру пропускную способность и стоимость:

- $\alpha(e) = \begin{cases} 2, & s \in e \\ 1, & s \notin e, \end{cases}$
- $c(e) = \begin{cases} \varphi(v, u, u'), & e = \{\{u, u'\}, v\}, \{u, u'\} \in D_1, v \in D_2 \\ 0, & \text{иначе.} \end{cases}$

Схематичное изображение полученной сети изображено на рисунке 6.1.

Функция $f : V \times V \to \mathbb{Q}$ называется *потоком*, если она удовлетворяет следующим условиям:

- $f(u, v) \leqslant \alpha(u, v)$ для любого ребра $\{u, v\}$,
- $f(u, v) = -f(v, u)$ для любого ребра $\{u, v\}$,
- $\sum\limits_{v \in N(u)} f(u, v) = 0$ для любой вершины $u \in V \backslash \{s, t\}$.

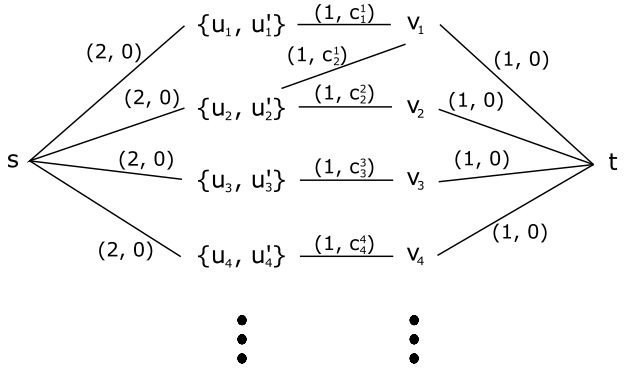

Рис. 6.1: Схематичное изображение сети, сопоставленной исходному графу. Каждому ребру сопоставлена пара чисел, первое из которых равняется пропускной способности, а второе — смоимости ребра. В данных обозначениях $c_i^j = \varphi(v_j, u_i, u_i')$.

Величиной потока по определению называется $\sum\limits_{v \in N(s)} f(s, v)$, стоимостью потока назовем $\sum\limits_{e \in E} |f(e)|c(e)$. Известно, что существуют полиномиальные алгоритмы поиска максимального потока минимальной стоимости, причем если $\alpha(e) \in \mathbb{N}$ для любого ребра e, то $f(e) \in \mathbb{Z}$ ([23]).

Лемма 6.9. Каждому потоку в построенной сети, для которого выполнено $f(e) \in \mathbb{Z} \quad \forall e \in E$, можно сопоставить псевдопериферийное семейство $\mathcal{S}$, причем так, что мощность полученного S равняется величине потока, а $\varphi(\mathcal{S})$ – его стоимости.

Доказательство. Построим семейство $\mathcal{S}$ следующим образом:

$$v \in S_{\{u,u'\}} \iff f(\{u, u'\}, v) = 1$$

Его псевдопериферийность легко проверяется. □

Лемма 6.10. Каждому минимальному периферийному семейству $\mathcal{S}$ можно сопоставить поток в построенной сети так, чтобы мощность S равнялась величине потока, а $\varphi(\mathcal{S})$ — его стоимости.

Доказательство. Построим поток следующим образом:

- $f(\{u, u'\}), v) = \begin{cases} 1, & v \in S_{\{u,u'\}} \\ 0, & v \notin S_{\{u,u'\}}, \end{cases}$

- $f(s, \{u, u'\})$ полагается равным величине потока, исходящего из $\{u, u'\}$ (т.к. $|S_{\{u,u'\}}| \leqslant 2$, то эта величина не превосходит пропускной способности),
- $f(v, t)$ полагается равным величине потока, входящего в вершину v (тут мы используем тот факт, что множества $S_{\{u,u'\}}$ не пересекаются).

Теперь все свойства легко проверяются. □

Утверждение 6.1. Минимальное периферийное множество находится за полиномиальное по числу вершин время.

Доказательство. Построим сеть по правилам, описанным выше. Найдем максимальный поток минимальной стоимости, по лемме 6.9 преобразуем его в семейство $\mathcal{S}$. Оно будет периферийным, потому что в противном случае по лемме 6.10 мы смогли бы найти еще больший поток. Также по лемме 6.10 оно будет минимальным, иначе мы смогли бы найти максимальный поток меньшей стоимости. □

7 Другие параметры

В этом разделе мы докажем, что для некоторых параметров существование ядра полиномиального размера влечет за собой вложение **coNP** $\subseteq$ **NP/poly**.

Определение 7.1. *Полиномиальным отношением эквивалентности* называется отношение эквивалентности на множестве Σ^*, такое что

- существует алгоритм, отвечающий на вопрос об эквивалентности за полиномиальное время,
- число классов эквивалентности для любого конечного множества S полиномиально по $\max\limits_{x \in S} |x|$.

Определение 7.2. Язык $K \subseteq \Sigma^*$ *кросс-композируется* в параметризованный язык $L \subseteq \Sigma^* \times \mathbb{N}$, если для него существует полиномиальный алгоритм, который по входу $(x_1, \cdots, x_p) \in (\Sigma^*)^p$, где $x_1, \cdots, x_p$ эквиваленты по какому-либо полиномиальному отношению эквивалентности, возвращает (x^*, k), причем

- $k \leqslant poly(\max_{i=1}^{p} |x_i| + \log p)$,
- $(x^*, k) \in L \iff \forall i \quad x_i \in K$.

Теорема 7.1 ([24])**.** Если NP-трудный язык $K \subseteq \Sigma^*$ кросс-композируется в параметризованный язык $L \subseteq \Sigma^* \times \mathbb{N}$, то у языка L нет ядра полиномиального размера, если только не выполнено вложение **coNP** $\subseteq$ **NP/poly**.

Утверждение 7.1. Если для произвольного графа G параметр $\theta(G)$ не превосходит $poly(\sigma(G))$ асимптотически, где $\sigma(G)$ — некоторый другой параметр, то существование ядра относительно параметра θ влечет за собой существование ядра относительно параметра σ.

Доказательство. Пусть $\theta \leqslant Q(\sigma)$ асимптотически. Заметим, что поскольку для достаточно больших x выполнено $Q(x) \leqslant 0$, то с некоторого момента $Q(x)$ монотонно возрастает. Поэтому асимптотически $Q(\theta(G)) \leqslant Q(\sigma(G))$. □

В этой главе мы будем работать с языком Γ:

$$G \in \Gamma \iff G \text{ — гамильтонов}$$

Утверждение 7.2. Вес оптимального цикла коммивояжера не может быть меньше чем $\sum\limits_{v \in V} \min\limits_{u \in N(v)} w(v, u)$.

Доказательство. Рассмотрим произвольный путь P, сопоставим каждой вершине v ребро e_v — то, по которому он вышел из вершины v (если коммияжер был в вершине v больше одного раза и таких ребер несколько, то выберем любое из них). Тогда

$$w(P) \geqslant \sum_{v \in V} w(e_v) \geqslant \sum_{v \in V} \min_{u \in N(v)} w(v, u).$$

□

Следствие 7.1. В невзвешенном графе длина оптимального цикла не меньше $|V|$.

Утверждение 7.3. Невзвешенный граф $G = (V, E)$ гамильтонов тогда и только тогда, когда длина оптимального цикла в нем равняется $|V|$.

Доказательство. Если существует гамильтонов цикл, то его длина в точности $|V|$ и по следствию 7.1 из утверждения 7.2 является минимальной. Обратно, цикл коммивояжера обязан содержать все $|V|$ вершин, следовательно, цикл длины $|V|$ содержит каждую вершину ровно по одному разу, поэтому является гамильтоновым. □

Также нам понадобится следующее определение:

Определение 7.3. Ребро $\{u, v\}$ называется *мостом*, если после его удаления число компонент связности в графе строго увеличивается.

Утверждение 7.4. Существует оптимальный путь, в котором мост проходится не более чем дважды. Более того, если вес моста строго положителен, то в любом оптимальном пути он проходится ровно дважды.

Доказательство. Следует непосредственно из утверждения 4.2 и того факта, что если $\{u, v\}$ — мост, то любой путь из вершины u в вершину v содержит ребро $\{u, v\}$. □

7.1 Ширина ленты

Определение 7.4 ([1])**.** *Шириной ленты* (по-английски *bandwidth*) графа будем называть следующую величину:

$$\min_{\substack{i:V\to\mathbb{N} \\ i \text{ инъективно}}} \max_{\{u,v\}\in E} |i(u) - i(v)|.$$

Теорема 7.2. Язык Γ кросс-композируется в язык **GTSP**, параметризованный шириной ленты.

Доказательство. Пусть даны невзвешенные графы $G_1, \cdots, G_p$. Произвольным образом выберем в каждом из графов вершины $v_1, \cdots, v_p$ соответственно. Построим на этих вершинах путь, объединив таким образом графы $G_1, \cdots, G_p$ в новый граф G (рис. 7.1). Заметим, что ширина ленты полученного графа не превосходит $2 \max\limits_i |V_i|$.

Действительно, пронумеруем вершины графа G_i ($i = 1, \cdots, p$) числами из отрезка $\left[1 + \sum\limits_{j=1}^{i-1} |V_i|; \sum\limits_{j=1}^{i} |V_i|\right]$. Такая нумерация задаст инъективное отображение φ, для которого выполнено:

- если $u, u' \in V_i$, то $|\varphi(u) - \varphi(u')| \leqslant |V_i|$
- $|\varphi(v_i) - \varphi(v_{i+1})| \leqslant |V_i| + |V_{i+1}|$.

Осталось показать, что выполнено

$$\forall i \in \{1, \cdots, p\} \quad G_i \in \Gamma \iff (G, \sum_{j=1}^{p} |V_j| + 2(p-1)) \in \textbf{GTSP}$$

Но это автоматически следует из того, что ребра $\{v_i, v_{i+1}\}$ $(i = 1, \cdots, p)$ являются мостами, поэтому проходятся дважды, а невзвешенный граф гамильтонов тогда и только тогда, когда длина оптимального пути равняется $|V|$ (утверждения 7.4 и 7.3). □

Следствие 7.2. Не существует ядра полиномиального рамера относительно ширины ленты, если только не выполнено вложение **coNP** $\subseteq$ **NP**/**poly**.

Доказательство. Следует из теоремы 7.1. □

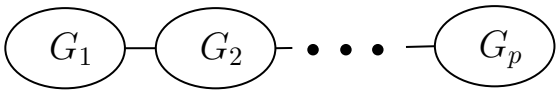

Рис. 7.1: Конструкция в теореме 7.2

7.2 Остовная высота

Определение 7.5 ([1])**.** Корневое дерево T назовем *разреженным остовным деревом* графа G, если G получается из T добавлением ребер $\{u, v\}$, где u — предок v. *Остовной высотой* (по-английски *treedepth*) графа G назовем минимальную высоту среди всех его разреженных остовных деревьев.

Теорема 7.3. Язык Γ кросс-композируется в язык **GTSP**, параметризованный остовной высотой.

Доказательство. Пусть даны невзвешенные графы $G_1, \cdots, G_p$. Выберем в каждом из них произвольным образом вершины $v_1, \cdots, v_p$ соответственно. Создадим новую

вершину v и соединим ее со всеми вершинами $v_1, \cdots, v_p$. Получим новый граф G (рис. 7.2), остовная высота которого не превосходит $\max\limits_i |V_i| + 1$.

Действительно, рассмотрим следующее остовное дерево графа G:

- корнем дерева назначим вершину v,
- если вершины u, u' принадлежат графу G_i, то одна из них является потомком другой,
- если вершины u, u' принадлежат различным графам G_i, G_j, то ни одна из них не является потомком другой.

Очевидно, что оно является разреженным остовным деревом графа G. С другой стороны, его высота есть в точности $\max\limits_i |V_i| + 1$.

Тот факт, что

$$\forall i \in \{1, \cdots, p\} \quad G_i \in \Gamma \iff (G, \sum_{j=1}^{p} |V_j| + 2(p-1)) \in \textbf{GTSP}$$

совершенно аналогично предыдущей теореме следует из утвержений 7.4 и 7.3. □

Следствие 7.3. Не существует ядра полиномиального размера относительно остовной высоты, если только не выполнено вложение **coNP** $\subseteq$ **NP**/**poly**.

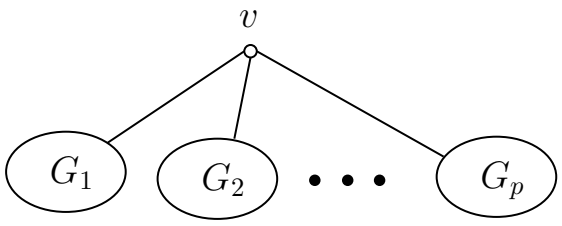

Рис. 7.2: Конструкция в теореме 7.3

7.3 Расстояние до клики

Определение 7.6 ([1])**.** *Расстоянием до клики* (по-английски *distance to clique*) называется минимально возможная мощность такого подмножества множества вер-

шин, при удалении которого оставшийся граф является полносвязным (т.е. любые две вершины которого соединены ребром).

Утверждение 7.5. Если после ограничения задачи коммивояжера на некоторое семейство графов с константным параметром θ она всё еще остается NP-трудной, то существование ядра относительно параметра θ влечет за собой равенство классов **P** и **NP**.

Доказательство. К любому графу из заданного семейства применим алгоритм кернелизации, который за полиномиальное время вернет ядро константного размера. Для полученного графа задачу можно будет решить полным перебором за константное время. □

Утверждение 7.6. Не существует ядра относительно таких параметров, как род поверхности, максимальная степень вершины и расстояние до клики, если $\mathbf{P} \neq \mathbf{NP}$.

Доказательство. Следует из утверждения 7.5 и того, что для задача коммивояжера на полном графе и задача коммивояжера на планарном 3-регулярном двудольном графе являются NP-трудными ([25]). □

Утверждение 7.7. Не существует ядра относительно таких параметров, как число независимости, обхват, хроматическое число, кликовое число, если только $\mathbf{P} \neq \mathbf{NP}$.

Доказательство. Следует из утверждения 7.1 и рис. 1. □

8 Заключение

В данной работе были предложены два алгоритма редукции данных, гарантирующие ядра полиномиального размера относительно таких параметров, как размер минимального разрезающего циклы набора ребер и размер минимального вершинного покрытия. Также была доказана невозможность существования ядер полиномиального размера относительно таких параметров как ширина ленты графа, максимальная степень, хроматическое число, остовная высота, расстояние до клики, число независимости, кликовое число, обхват.

Открытым вопросом остается существование ядра полиномиального размера относительно размера минимального разрезающего циклы набора вершин.

9 Список литературы

[1] Manuel Sorge и Mathias Weller. «The Graph Parameter Hierarchy». В: февр. 2019. url: https://manyu.pro/assets/parameter-hierarchy.pdf.

[2] Karp R.M. «Reducibility Among Combinatorial Problems». В: *Complexity of Computer Computations. The IBM Research Symposia Series.* (1972). doi: 10.1007/978-1-4684-2001-2_9.

[3] Stephen A. Cook. «The Complexity of Theorem-Proving Procedures». В: STOC '71 (1971), с. 151—158. doi: 10.1145/800157.805047.

[4] Левин Л.А. «Универсальные задачи перебора». В: *Проблемы передачи информации* 9 (3 1973), с. 115—116.

[5] Christos H. Papadimitriou и Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, 1998.

[6] D. J. Rosenkrantz, R. E. Stearns и P. M. Lewis. «Approximate algorithms for the traveling salesperson problem». В: (1974), с. 33—42. doi: 10.1109/SWAT.1974.4.

[7] Sanjeev Khanna, Nathan Linial и Shmuel Safra. «On the Hardness of Approximating the Chromatic Number». В: *Combinatorica* 20.3 (март 2000), с. 393—415. issn: 1439-6912. doi: 10.1007/s004930070013.

[8] Sartaj Sahni и Teofilo Gonzalez. «P-Complete Approximation Problems». В: *J. ACM* 23.3 (июль 1976), с. 555—565. issn: 0004-5411. doi: 10.1145/321958.321975.

[9] J. B. Robinson. «On the Hamiltonian game (a traveling-salesman problem).» В: (1949).

[10] René van Bevern and Viktoriia A. Slugina. «A historical note on the 3/2-approximation algorithm for the metric traveling salesman problem». В: *Historia Mathematica* 53 (2020), с. 118—127. issn: 0315-0860. doi: https://doi.org/10.1016/j.hm.2020.04.003.

[11] Nicos Christofides. *Worst-case analysis of a new heuristic for the travelling salesman problem*. Тех. отч. Carnegie-Mellon Univ Pittsburgh Pa Management Sciences Research Group, 1976.

[12] А.И. Сердюков. «О некоторых экстремальных обходах в графах». В: *Управляемые системы* (17 1978), с. 76—79.

[13] Anna Karlin, Nathan Klein и Shayan Oveis Gharan. «A (Slightly) Improved Approximation Algorithm for Metric TSP». В: (июль 2020).

[14] M. Dorigo. «Optimization, Learning and Natural Algorithms». В: 1992.

[15] Dervis Karaboga. «An Idea Based on Honey Bee Swarm for Numerical Optimization, Technical Report — TR06». В: *Technical Report, Erciyes University* (янв. 2005).

[16] Robert Reynolds. «Evolutionary Computation: The Fossil Record. David Fogel, IEEE PRESS, 1998, ISBN 0-7803-3481-7». В: *Biosystems* 56 (май 2000), с. 145—146. doi: 10.1016/S0303-2647(00)00075-7.

[17] John D. C. Little и др. «An Algorithm for the Traveling Salesman Problem». В: *Oper. Res.* 11.6 (дек. 1963), с. 972—989. issn: 0030-364X. doi: 10.1287/opre.11.6.972.

[18] Ralph Gomory. «Outline of an Algorithm for Integer Solutions to Linear Programs and An Algorithm for the Mixed Integer Problem». В: *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art* (янв. 2010), с. 77—103. doi: 10.1007/978-3-540-68279-0_4.

[19] César Rego и др. «Traveling salesman problem heuristics: Leading methods, implementations and latest advances». В: *European Journal of Operational Research* 211.3 (2011), с. 427—441. issn: 0377-2217. doi: https://doi.org/10.1016/j.ejor.2010.09.010.

[20] Fedor V. Fomin и др. *Kernelization: Theory of Parameterized Preprocessing*. Cambridge University Press, 2019. doi: 10.1017/9781107415157.

[21] A. Frank и E. Tardos. «An application of simultaneous Diophantine approximation in combinatorial optimization». В: *Combinatorica* 7.1 (март 1987), с. 49—65. issn: 1439-6912. doi: 10.1007/BF02579200.

[22] Michael Etscheid и др. «Polynomial kernels for weighted problems». В: *Journal of Computer and System Sciences* 84 (2017), с. 1—10. issn: 0022-0000. doi: https://doi.org/10.1016/j.jcss.2016.06.004.

[23] Jack Edmonds и Richard M. Karp. «Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems». В: *J. ACM* 19.2 (апр. 1972), с. 248—264. issn: 0004-5411. doi: 10.1145/321694.321699.

[24] Holger Dell. «AND-Compression of NP-Complete Problems: Streamlined Proof and Minor Observations». В: *Algorithmica* 75 (2016), с. 403—423.

[25] T. Akiyama, Takao Nishizeki и N. Saito. «NP-Completeness of the Hamiltonian Cycle Problem for Bipartite Graphs». В: *Journal of Information Processing* 3 (1980), с. 73—76.