ИНФОРМАТИКА И ЕЁ ПРИМЕНЕНИЯ, 2017. Т. 11. Вып. 1. С. 79–89

АЛГОРИТМ ПРЕОБРАЗОВАНИЯ ОДНОГО ГРАФА В ДРУГОЙ С МИНИМАЛЬНОЙ ЦЕНОЙ*

К. Ю. Горбунов[1], В. А. Любецкий[2]

Аннотация: Рассматриваются ориентированные графы, состоящие из любого числа дизъюнктных цепей и циклов, ребрам графов приписаны без повторений их имена — натуральные числа. Фиксирован список операций, каждая из которых по-своему преобразует один граф в другой, ей приписано число — цена данной операции. Нужно найти минимальную по суммарной цене последовательность операций, которая для двух данных графов преобразует один в другой. Эта задача самым широким образом применяется в прикладных вопросах. По-видимому, она является NP-трудной и поэтому может быть эффективно решена только при том или ином условии на цены или при некотором ограничении на графы. Ее решение при достаточно широких условиях получено в виде линейных по времени и памяти алгоритмов, для которых доказана точность (неэвристичность), т. е. доказано, что они всегда находят минимальную по цене последовательность операций. Задача давно решается многими эвристическими алгоритмами, которые тестировались на разных данных, но предлагаемые авторами решения — первые среди точных.

Ключевые слова: ориентированный граф из цепей и циклов; преобразование графов с минимальной ценой; точное линейное решение; условие на графы; условие на цены; условно кратчайшее решение

DOI: 10.14357/19922264170107

1 Введение. Постановка задачи

1.1 CC-графы

В работе рассмотрена следующая комбинаторно-оптимизационная задача. Назовем *CC-графом* ориентированный граф, состоящий из любого числа дизъюнктных цепей и циклов, включая петли, ребрам которого приписаны без повторений натуральные числа (*имена* ребер). Цепи и циклы являются (без учета ориентации) компонентами связности такого графа, которые будем называть *компонентами*. Фиксирован список операций, которые преобразуют один CC-граф в другой такой же граф.

Можно рассматривать более общий случай графов и любой список операций, но в длительной истории исследования этой задачи (по разным, прежде всего прикладным, причинам) сформировалось указанное определение графа и тот список операций, который приведен в подразд. 1.2. Каждой операции приписано число, которое называется ее *ценой*. В прикладных задачах цены являются строго положительными рациональными числами, но, разумеется, теоретически их можно считать натуральными числами. Любой последовательности операций, применяемых друг за другом, начиная с данного CC-графа a и заканчивая некоторым результирующим CC-графом b, приписывается *суммарная цена* — сумма цен всех операций в этой последовательности.

Итак, пусть даны два CC-графа a и b. Требуется найти минимальную по функционалу суммарной цены последовательность операций, которая преобразует a в b. Такую последовательность называют *кратчайшей*, а ее цену — *кратчайшей ценой*. Предполагается, хотя это не доказано, что задача нахождения кратчайшей последовательности или кратчайшей цены для переменных a, b и переменных цен операций является NP-трудной. Задача остается таковой, если фиксировать произвольные (случайные) цены операций. Поскольку практический интерес представляют линейные или, во всяком случае, полиномиальные алгоритмы низкой степени, для их поиска приходится накладывать условия на соотношение цен или на вид графов. В части второго популярно такое ограничение: в последовательности операций, которая преобразует a в b, включая и сами a и b, присутствует один и тот же постоянный набор имен. Задачу с этим ограничением назовем задачей с *постоянным составом* (имен ребер). В отсутствие этого ограничения задачу назовем задачей с *переменным составом*. В разд. 2 приводится схема линейного по времени и памяти

*Работа выполнена при финансовой поддержке Российского научного фонда (проект 14-50-00150).

[1]Институт проблем передачи информации им. А. А. Харкевича Российской академии наук, gorbunov@iitp.ru

[2]Институт проблем передачи информации им. А. А. Харкевича Российской академии наук; механико-математический факультет Московского государственного университета им. М. В. Ломоносова, lyubetsk@iitp.ru

алгоритма ее решения в трех случаях: два относятся к постоянному составу и один к переменному составу. Все нюансы работы алгоритма, сопровождаемые рисунками, а также детали доказательств приведены в [1, 2].

1.2 Операции над СС-графами

Фиксируются следующие операции, называемые *стандартными*:

- разрезать две вершины, имеющиеся в графе, и по-новому отождествить (склеить) четыре образовавшихся края (*двойная переклейка*) (рис. 1, *а*);
- разрезать вершину и по-новому отождествить (склеить) один образовавшийся край с каким-то свободным краем в графе (*полуторная переклейка*) (рис. 1, *б*);
- разрезать вершину или отождествить два свободных края (одинарные переклейки — *разрез* и *склейка*) (рис. 1, *в*).

Также фиксируются две *дополнительные операции*, называемые соответственно *вставкой* и *удалением*: добавить/удалить цепь X в граф или из графа (рис. 2).

Для вставки это означает: добавить в граф саму цепь X или ее же с отождествлением ее концов, т. е. добавить новую цепь или новый цикл; или добавить цепь X в цепь, уже имеющуюся в графе, вместо ее конца или ее внутренней вершины; аналогично для вставки в цикл. И симметрично для операции удаления. При этом добавлять можно цепь, имена которой содержатся в $b\backslash a$, а удалять можно цепь, имена которой содержатся в $a\backslash b$. В [3] предложенный авторами алгоритм был применен в конкретном прикладном исследовании, и там содержатся более подробные, чем в подразд. 1.3, сведения по истории задачи, однако содержание работы [3] не используется в данной статье.

1.3 Обзор непосредственно примыкающей литературы

Приведем несколько математических результатов других авторов по этой задаче. Таких результатов немного. В случае *постоянного* состава и *различных* цен алгоритм решения с его математическим обоснованием не был предложен. В работах [4, 5] рассматривается сразу *переменный состав*. В [4] на цены накладывается условие: у всех стандартных операций они равны 1, а у операций вставки и удаления равны и не больше 1. В [5] на це-

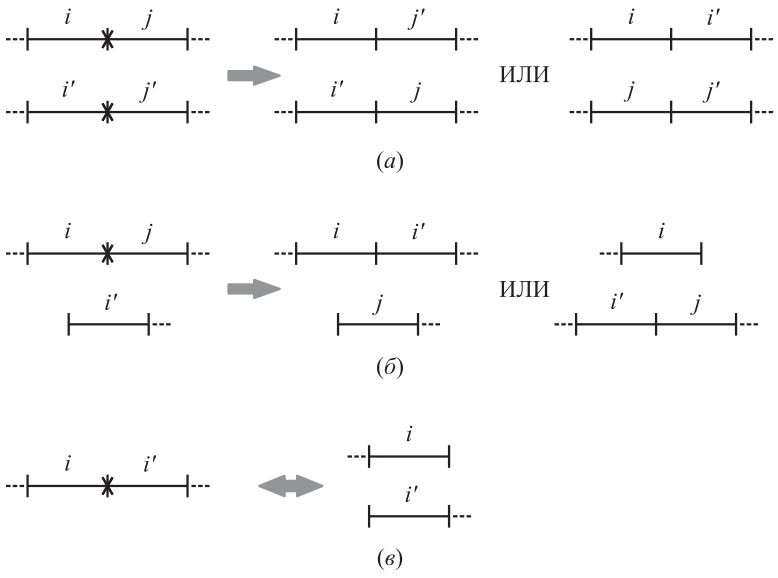

Рис. 1 Четыре стандартные операции над СС-графом: (*а*) двойная переклейка; (*б*) полуторная переклейка; (*в*) разрез и склейка

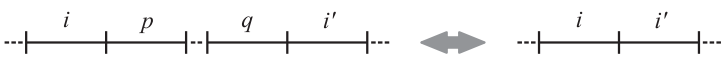

Рис. 2 Две дополнительные операции над СС-графом: удаление и вставка

ны накладывается другое, более слабое условие — опущено «не больше 1», но зато рассматриваются СС-графы только из циклов. По существу, последнее условие эквивалентно тому, что из стандартных операций допускается только двойная переклейка. Насколько авторы понимают, результат в [4] не содержит корректных доказательств, так что указанный случай не получил обоснования. Подходы, которые предложены в них (т. е. определения вспомогательных графов — прием, который используется во всех работах на эту тему, включая и данную), отличаются от предлагаемого авторами статьи.

В [2] авторы описали линейный по времени и памяти алгоритм построения кратчайшей последовательности операций, преобразующих один СС-граф a в другой СС-граф b для случая *переменного* состава, если цены всех операций равны. Если все цены равны, кратчайшую последовательность называют *минимальной*, а кратчайшую цену — *минимальной ценой*. Конечно, главный интерес представляет общий случай задачи, в котором цены не равны.

Как и в [2], в разд. 2 изложение строится по такому плану. Определяется понятие общего графа $a + b$, иное, нежели в [4, 5]; показано, что исходная задача эквивалентна приведению $a + b$ к специальному виду, который называется *финальным*, аналогами операций, которые определены в подразд. 1.2. Затем описывается алгоритм и приводится доказательство его точности (неэвристичности, корректности), т. е. доказательство того, что он действительно находит минимум суммарной цены. Как отмечалось выше, некоторые технически громоздкие детали в описании и доказательстве приведены в [1]. Вместо «неэвристический алгоритм» или «неэвристическое решение» иногда говорят *точный алгоритм* или *точное решение* соответственно.

В исследованиях этой задачи используется различная терминология. Так, в [2] СС-граф называется *структурой*; для согласования с этой работой будем далее использовать последний, более привычный термин. В [3] и в других работах СС-граф (структура) называется хромосомной структурой, его ребра называются генами, а компоненты — хромосомами. Это связано с тем, что задача возникла в контексте биоинформатики, где для ее решения разработано большое число более или менее эвристических алгоритмов. В [3] приводится краткий обзор по истории исследований задачи.

Напомним, что минимальная последовательность и минимальная цена являются частными случаями кратчайшей последовательности и кратчайшей цены.

2 Решение задачи

2.1 Общий граф и идея алгоритма

Общий граф $a + b$ двух структур a и b имеет следующие вершины. *Обычные* вершины — имена краев одноименных ребер в a и b; например, начало ребра с именем 3 будет иметь имя 3_1. И *особые вершины* — максимальные по включению связные участки из ребер, принадлежащих лишь одной из структур, которые называют *блоками*. Блок принадлежит одной из структур, и соответствующая особая вершина помечается как a- или b-вершина. Ребра общего графа следующие. *Обычное* ребро соединяет две обычных вершины, если соответствующие им края отождествлены (склеены) в a или в b. А *особое* ребро соединяет обычную вершину с особой, если в a или в b край, соответствующий обычной вершине, отождествлен (*склеен*) с краем блока, соответствующего особой вершине. Такое ребро помечается как a- или b-ребро. *Петля* в $a + b$ соответствует циклу, который является блоком; иными словами, особая вершина этого блока соединяется с собой. *Висячим* называется ребро, инцидентное особой вершине степени 1. Пример двух структур и их общего графа приведен в [3] на рис. 1 и 2. Таким образом, общий граф несет информацию о склейках одновременно в a и в b.

Общий граф — неориентированный, он состоит из связных компонент — также цепей и циклов. Невисячие особые ребра присутствуют в нем парами — ребрами, инцидентными одной особой вершине; такую пару удобно считать за одно двойное ребро. Поэтому *размером компоненты* назовем сумму в ней числа обычных ребер с половиной числа особых невисячих ребер (в [2] эта величина названа длиной компоненты, что вызывает путаницу с обычной длиной цепи или цикла). Для изолированных обычных вершин и петель размер считаем равным 0, для изолированных особых вершин (не петель) — равным -1. Общий граф называется *финального вида*, если каждая его компонента — изолированная обычная вершина или цикл без особых ребер размера (или в данном случае то же самое — длины) 2, одно ребро из a и другое из b.

Эти определения приведены в [2, 3] и частично в [6] и здесь повторяются для удобства читателя.

В [2] доказано: для любых структур a и b существует кратчайшая последовательность, в которой все удаления предшествуют всем вставкам и все операции сохраняют блоки без изменения. Хотя там считалось, что цены всех операций равны, легко проверить, что приведенное доказательство сохраняется без изменения, если равны только цены

стандартных операций. Действительно, в том доказательстве произвольная кратчайшая последовательность преобразуется в последовательность указанного вида, при этом стандартная операция переходит в стандартную, удаление — в удаление, вставка — во вставку. Поэтому как там, так и здесь суммарная цена (иногда будем говорить — цена) последовательности не меняется.

Из этого *утверждения* следует: в предположении одинаковых цен стандартных операций задача поиска кратчайшей последовательности для структур a и b эквивалентна задаче приведения этих структур к какой-нибудь одной структуре c двумя суммарно кратчайшими последовательностями, причем вставка не используется, а цена удаления, применяемого в преобразованиях b, равна цене вставки.

Действительно, если c — структура в кратчайшей последовательности, полученная после выполнения всех удалений и до всех вставок, то можно преобразовать к ней структуру a (прямыми операциями) и структуру b (обратными операциями). А последняя задача эквивалентна задаче преобразования общего графа $a + b$ к финальному виду $c + c$ следующими аналогами исходных операций (рис. 3).

Двойная переклейка (рис. 3, *а*): удаление двух одинаково помеченных ребер общего графа и соединение четырех образовавшихся концов двумя новыми неинцидентными ребрами с той же пометкой. Если при этом образуется ребро с особыми концами (оба относятся к a или оба к b), то оно заменяется одной особой вершиной, которой приписано объединение блоков двух исходных особых вершин.

Полуторная переклейка (рис. 3, *б*): удаление ребра общего графа и соединение ребром с той же пометкой, скажем a, одного из его концов с обычной

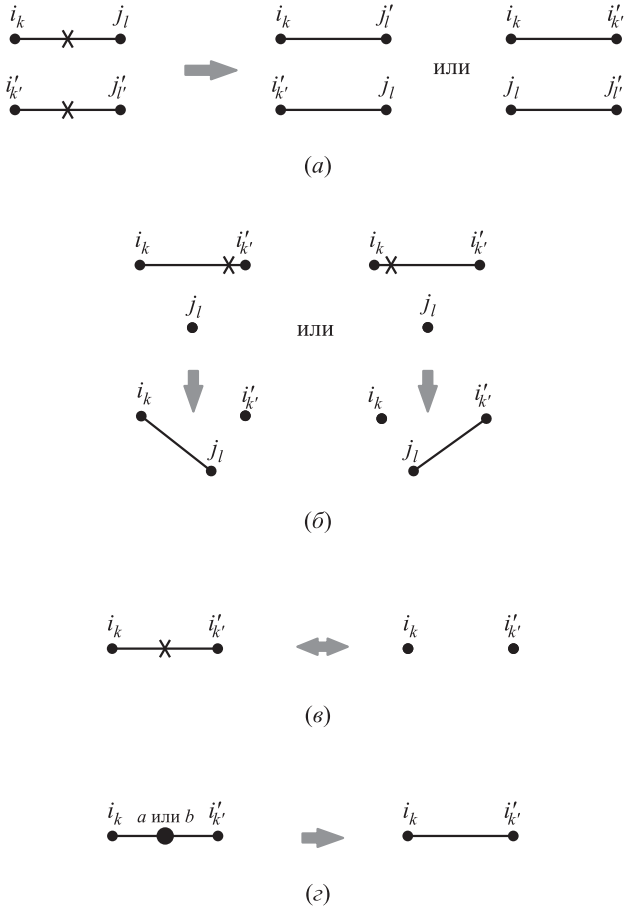

Рис. 3 Операции, разрешенные над общим графом: (*а*) двойная переклейка; (*б*) полуторная переклейка; (*в*) разрез и склейка; (*г*) a- или b-удаление. Большой кружок показывает особую вершину. Отметим: операция вставки оказывается ненужной, но зато операция удаления применяется в двух вариантах: a-удаления и b-удаления особой вершины; первая — по цене удаления, вторая — по цене вставки

вершиной, не инцидентной ребру с этой пометкой, или с особой вершиной степени не больше 1 с той же пометкой (с возможным последующим отождествлением двух особых вершин).

Склейка (рис. 3, *в*): добавление ребра (скажем, с пометкой a) между вершинами, каждая из которых является или обычной, не инцидентной ребру с пометкой a, или особой, степени не больше 1, с той же пометкой (с возможным последующим отождествлением двух особых вершин).

Разрез (рис. 3, *в*): удаление любого ребра.

Удаление особой вершины (рис. 3, *г*): если ее степень 2, то она удаляется и инцидентные ей ребра склеиваются в одно ребро с той же пометкой; если ее степень 1, то она удаляется вместе с инцидентным ей ребром; если ее степень 0 или это петля, то вершина с петлей удаляются. Удаление особой a-вершины получает цену операции удаления, удаление особой b-вершины — цену операции вставки. Условимся далее в выражении особая a- или b-вершина опускать слово «особая».

Забегая вперед, опишем *идею предлагаемого алгоритма*. В случае *постоянного* состава в общем графе имеются лишь обычные вершины и ребра. Алгоритм приводит его к финальному виду в два этапа. Первый этап — двойными переклейками разбить все циклы на циклы длины 2. Второй этап — обработка цепей: если цена двойной переклейки меньше цены полуторной, замкнуть цепи в циклы и обработать их, как на этапе 1. Иначе нужно полуторными переклейками от цепи пошагово отщеплять циклы длины 2.

В случае *переменного* состава алгоритм из компонент вырезает обычные ребра, замыкая их в циклы длины 2. Затем обрабатываются цепи. Это связано с тем, что совместная обработка цепей позволяет экономить число операций по сравнению с тем их числом, которое получилось бы при обработке каждой цепи в отдельности; это — основная идея алгоритма. Совместная обработка цепей описана в [2], где доказано, что она приводит к максимально возможной экономии числа операций, т. е. к минимизации числа операций.

Ниже в описании алгоритма совместная обработка цепей выполняется на шаге 3, каждый пункт которого соответствует определенной совместной обработке (можно сказать — взаимодействию) цепей. После шага 3 общий граф может еще содержать цепи, а также в нем остаются исходные циклы. Между этими компонентами уже невозможны взаимодействия, которые экономили бы число операций. Но возможны взаимодействия, заменяющие дорогое удаление b-вершины на операцию с меньшей ценой. Эти взаимодействия описаны на шаге 4 алгоритма.

2.2 Приведение общего графа в случае постоянного состава и разных цен операций

Рассмотрим случай постоянного состава, вставки и удаления отсутствуют, цены операций различны. Точнее, предполагается, что цены операций удовлетворяют одному из двух условий: $c_2 \le c_1 \le \le c'_1 \le c_{1,5}$ (*циклический* вариант) и $c_1 \le c'_1 \le \le c_{1,5} \le c_2$ (*линейный* вариант). Здесь указаны соотношения между ценами разреза c_1, склейки c'_1, полуторной переклейки $c_{1,5}$, двойной переклейки c_2.

В этом случае предлагается решение несколько **измененной задачи**: кратчайшая последовательность ищется среди всех минимальных последовательностей. Она называется *условно кратчайшей*. Решение задачи в этом смысле будем называть *условной оптимизацией*. Авторы не знают, существует ли полиномиальный по времени алгоритм решения безусловной задачи даже при одном из указанных соотношений цен, если только не все цены равны. Конечно, если они равны, то *условная* оптимизация совпадает с *безусловной*, т. е. с решением исходной задачи.

В рассматриваемом случае общий граф состоит из циклов и цепей, в которых чередуются a- и b-ребра. В случае постоянного состава *качеством* $H(a+b)$ общего графа $a+b$ назовем число циклов, сложенное с половиной числа четных цепей в нем. Четной называется цепь с четным числом ребер, а также цепь размера ноль; цепи нечетного размера не учитываются; понятия размера и длины в этом пункте совпадают. Пусть структуры a и b имеют по n ребер. Для построения минимальной последовательности решающее значение имеет возрастание качества от значения $H(a+b)$ до значения n на $+1$ при выполнении каждой операции; таким образом, минимальную длину можно указать сразу: она равна $n - H(a+b)$.

Лемма 1.

1. *Каждая стандартная операция изменяет качество общего графа на* 0 *или* ± 1.

2. *Для нефинального графа существует операция, увеличивающая его качество на* 1.

3. *Граф* $a+b$ *финальный, если и только если* $a = b$; *для финального графа* $a+b$ *выполняется* $H(a+b) = n$.

4. *Как безусловная, так и условная задачи для* a *и* b *эквивалентны соответствующим задачам о приведении общего графа* $a+b$ *к финальному виду*.

5. *Существует последовательность операций, преобразующая $a+b$ к финальному виду, на каждом шаге которой качество увеличивается ровно на* 1*; ее длина равна $k = n - H(a+b)$.*

6. *Минимальная длина равна k.*

7. *Минимальными последовательностями для $a+b$ являются в точности те, у которых каждая операция увеличивает качество общего графа на* 1*. Их длины равны k.*

Простое доказательство леммы 1 приведено в [1, п. 3].

Опишем точный линейный алгоритм приведения к финальному виду в случаях циклического и линейного соотношений цен. Пункт 3 в [1] содержит рисунки, наглядно поясняющие его работу. Для **циклического варианта** он состоит из трех шагов.

Шаг 1. Если имеется цикл длины, строго большей двух, двойной переклейкой разбиваем его на два цикла, один из которых имеет длину 2.

Шаг 2. Склейкой каждую нечетную цепь замыкаем в цикл, после чего применяем шаг 1.

Шаг 3. Полуторной переклейкой каждую ненулевую четную цепь замыкаем в цикл, один край цепи становится нулевой цепью. Затем применяем шаг 1.

Алгоритм решения условной задачи для **линейного варианта** состоит также из трех шагов.

Шаг 1. Тот же, что и в предыдущем алгоритме.

Шаг 2. Разрезом от каждой нечетной цепи отделяем крайнюю вершину, получаем четную цепь на 1 меньшей длины и нулевую цепь.

Шаг 3. Полуторной переклейкой каждую ненулевую четную цепь укорачиваем на 2 ребра и замыкаем два ее крайних ребра в цикл, пока в общем графе не останется ненулевых цепей.

Если ни один шаг не применим, то общий граф уже имеет финальный вид и к нему применяется пустая последовательность операций. □

Отметим: от [4, 5] приведенное ниже доказательство теорем 1 и 2 отличается другими условиями на цены, использованием другого вспомогательного графа, меньшего размера, и индукцией по величине $C(G)$ общего графа, которая и составляет суть приводимых доказательств (не говоря об отсутствии полного доказательства в [4, 5]).

Теорема 1. *Указанные линейные алгоритмы точно решают задачу условной оптимизации для циклического и линейного вариантов цен.*

Схема доказательства. Минимальность полученной последовательности следует из леммы 1. Из нее же следует линейность алгоритма по времени.

Докажем, что полученная последовательность — кратчайшая. Для этого выразим суммарную цену $c(G)$ в полученной алгоритмом последовательности через числовые характеристики графа G (подробности приведены в [1, п. 3]). Кратчайшую цену для приведения графа G к финальному виду *обозначим* $C(G)$. Индукцией по величине $C(G)$ покажем, что для всех графов G выполняется неравенство $c(G) \le C(G)$. Отсюда $c(G) = C(G)$, что и требуется. Число вершин в графе G фиксировано, поэтому множество минимальных цен конечно. Индукция идет по естественному порядку в этом множестве цен. Если $C(G) = 0$, то граф G финального вида и $c(G) = 0$.

Индуктивный шаг. Пусть для всех графов G', у которых $C(G') < C(G)$, выполняется неравенство $c(G) \le C(G)$. Докажем его для G. Рассмотрим приводящую последовательность для G. Обозначим через o ее первую операцию, $c(o)$ — ее цену, $o(G)$ — результат ее применения к G. Достаточно проверить неравенство $c(o') \ge c(G) - c(o'(G))$ для каждой операции o'. Действительно, по предположению индукции имеем: $c(o(G)) \le C(o(G))$. Отсюда $c(G) \le c(o(G)) + c(o) \le C(o(G)) + c(o) = C(G)$. Подробности проверки приведены в [1, п. 3]. □

2.3 Приведение общего графа в случае переменного состава и разных цен операций

Дан общий граф $a+b$ и число ε, $0 \le \varepsilon \le 1$. Разрешены все операции, т. е. в силу утверждения в подразд. 1.1 стандартные операции, удаления a- и b-вершин (особых вершин с пометкой a или b). Пусть цены стандартных операций и a-удаления равны 1, а цена b-удаления вершины равна $1+\varepsilon$. Термин *конец*, естественно, относится к концу ребра или к изолированной вершине в общем графе.

Предлагаемый алгоритм компьютерно тестировался в общем случае, если цена b-удаления больше цены всех других операций. Как правило, алгоритм находил ответ, близкий к кратчайшей последовательности. Эта более общая ситуация здесь не рассматривается, но с учетом возможного эвристического использования в описание алгоритма, которое приведено ниже, включены соответствующие пояснения; они не используются в доказательстве, которое также приводится ниже.

Краткое описание алгоритма

Шаг 1. Удалить особые a-петли.

Шаг 2. Вырезать все обычные ребра, не входящие в 2-циклы (т. е. циклы размера 2), замыкая их в финальные 2-циклы двойной (если ребро не крайнее) или полуторной (если оно крайнее) переклейками или склейкой (если оно изолированное). В [1, п. 4] содержится подробное описание работы алгоритма на шагах 2 и 3 с рисунками.

Шаг 3. Фактически этот шаг тот же, что и в [2] (более подробно он описан в [1, п. 4]). Напомним его смысл. В множестве цепей общего графа (после шага 2) выделяются небольшие попарно непересекающиеся подмножества мощности от 2 до 4. Внутри каждого подмножества M производится $(|M|-1)$ операций между цепями (взаимодействий) так, что если каждую цепь из M приводить к финальному виду автономно (т. е. без взаимодействий с другими компонентами), то число требуемых операций будет строго больше числа операций, требуемых, если сначала провести данное взаимодействие. Доказывается, что описанное множество взаимодействий дает максимально возможную экономию числа операций.

Шаг 4. На этом шаге в определенном порядке производятся взаимодействия между связными компонентами общего графа. Каждое взаимодействие производится до тех пор, пока есть компоненты, которые могут служить его аргументами. Эти взаимодействия не уменьшают общее число операций (точнее, сохраняют его), но позволяют заменить «дорогую» операцию удаления b-вершины на другую, более дешевую операцию. Например, если удалить две b-петли по отдельности, будет произведено две операции удаления b-вершины, если же сначала двойной переклейкой слить эти две петли в одну (это частный случай взаимодействия 4.1 из [1]), то одно удаление заменится на двойную переклейку. Подробно шаг 4 описан в [1, п. 4].

Шаг 5. Удаляем изолированные особые вершины и петли. Из оставшихся цепей удаляем особые вершины. Из циклов размера, большего 2, вырезаем 2-циклы так, чтобы происходило отождествление двух b-вершин (соответственно, в 2-цикл включается a-вершина). Из 2-циклов удаляем особые вершины.

Конец описания алгоритма. □

Докажем теорему о минимальности суммарной цены последовательности операций, которая получается в алгоритме, т. е. о точности (корректности) алгоритма.

Пусть B' — число циклов в графе $a+b$, содержащих b-вершину, но не содержащих a-вершину (назовем их b-циклами). Напомним обозначения из [2]: B — число особых вершин в $a+b$; S — сумма целых частей половин числа ребер (назовем число *длиной*) максимальных отрезков (*сегментов*) в $a+b$, которые состоят из обычных ребер, плюс число нечетных (т. е. нечетной длины) крайних сегментов минус число циклических сегментов. *Крайним* называется сегмент, расположенный с краю цепи, включая и случай целой цепи. Обычной называется пара, состоящая из одной из стандартных операций вместе с ее аргументом, результат которой не меняет число особых вершин. Далее «обычная» относится к операции, а ее аргумент подразумевается заданным. *Дефект* цепи (или цикла) равен минимальному числу обычных операций в последовательности, которая приводит ее (или его) к финальному виду, не считая вырезания обычных ребер на шаге 2; в последовательности могут встречаться и операции с их аргументами, которые не являются обычными; назовем их *особыми*. В [2] приведена зависимость дефекта от типа компоненты. Обозначим D сумму дефектов компонент графа $a+b$. Обозначим P разность величин D, вычисленных до и после применения шага 3 алгоритма. Заметим, что в любой последовательности операций, финализирующих общий граф, число особых операций равно числу особых вершин в нем, так что экономия числа операций может относиться лишь к обычным операциям. Поскольку все операции на шаге 3 особые, величина P равна числу операций, сэкономленных на шаге 3. Величина ε определена выше. Пусть $C = B + S + D - P + \varepsilon(B'+1)$.

Теорема 2. *Алгоритм строит последовательность операций, суммарная цена которой равна одному из трех значений $C-\varepsilon$, C, $C+\varepsilon$. Минимально возможная суммарная цена последовательности операций, приводящей граф $a+b$ к финальному виду, также равна одному из этих значений. Время работы алгоритма линейное по порядку.*

В доказательстве теоремы будут использованы нижеследующие леммы 2 и 3.

Лемма 2. *После выполнения шага 4 остается 0, 1 или 2 связных компоненты, имеющих b-вершину и не являющихся исходными b-циклами.*

Простое доказательство леммы приведено в [1, п. 4] (там это лемма 3).

Лемма 3. *Число обычных операций в алгоритме равно $S+D-P$.*

Доказательство. Напомним [2], что минимальное число обычных операций, требуемых для

приведения компоненты (после шага 2) к финальному виду без использования других компонент, равно ее дефекту. Настоящий алгоритм отличается от описанного в [2] наличием шага 4. Любая операция шага 4 либо особая и не меняющая дефект результата по сравнению с суммарным дефектом аргументов, либо обычная и уменьшающая его на 1. Поэтому обычных операций в алгоритме столько же, сколько и раньше, т. е. $S + D - P$. □

Доказательство теоремы 2. На шаге 5 для каждой компоненты, имеющей b-вершины, применяется ровно одна операция удаления b-вершины. По лемме 2 общее число таких операций равно $B' + n$, где n равно 0, 1 или 2. Всего особых операций B. В силу леммы 3 суммарная цена операций алгоритма равна $(1 + \varepsilon)(B' + n) + (B - B' - n) + + (S + D - P) = B + S + D - P + \varepsilon(B' + n)$, откуда следует первое утверждение теоремы.

Второе утверждение теоремы докажем индукцией по минимальной суммарной цене M операций, приводящих общий граф к финальному виду; имеется лишь конечное число возможных значений M на любом ограниченном отрезке, которые рассматриваем по их возрастанию. Рассуждая так же, как и в доказательстве теоремы 1, видим, что достаточно для любой операции o, примененной к произвольному общему графу G, проверить, что цена $c(o)$ операции o не меньше $C(G) - C(o(G))$, где $C(G)$ — величина C, определенная в формулировке теоремы 2. Подробности проверки приведены в [1, п. 4]. □

Следствие. Цена последовательности операций, которую строит описанный алгоритм, отличается от цены кратчайшей последовательности не более чем на ε.

Доказательство приведено в [7]. □

3 Обобщение: задача с повторением имен

3.1 Постановка задачи

Важное в прикладных вопросах обобщение рассмотренной выше задачи состоит в том, что в структурах разрешается повторение имен. Это обобщение назовем *задачей с повторениями* (или по историческим причинам говорят: задачей с паралогами). Пусть a и b — такие структуры. Например, имеются в a три ребра с именем k и в b два ребра с тем же именем k. Нужно найти биекцию меньшего из этих двух множеств ребер в большее (для данного имени k); и аналогично для каждого имени k, если ему соответствуют два таких множества, одно в a и другое в b, с разным числом элементов. Итак, нужно найти семейство биекций, индексированное k, при котором кратчайшая цена достигает минимального значения. Более детально на этом примере: нужно приписать этим пяти ребрам индекс i к их имени k (получатся *полные имена*, которые имеют вид $k.i$, где i меняется от 1 до 3) так, чтобы с новыми именами у всех повторяющихся ребер достигалось минимальное значение кратчайшей цены кратчайшего преобразования a в b. Индекс i определяет частичное соответствие между бывшими одноименными ребрами в a и b и, в частности, определяет, какие ребра общие и какие особые для этих структур. Например, эти три ребра можно индексировать $k.1$, $k.2$, $k.3$, а два других ребра индексировать $k.2$ и $k.3$, тогда ребро $k.1$ особое, а остальные ребра общие. Полные имена позволяют перейти от *задачи с повторениями к задаче без повторений* (имен), последняя рассматривалась в разд. 1 и 2.

В силу NP-трудности задачи с повторениями, нельзя найти точный полиномиальный алгоритм ее решения. Однако ниже будет показано, как математически строго свести ее к задаче целочисленного линейного программирования (ЦЛП). Как известно, для задач ЦЛП доступны компьютерные программы, выдающие, как правило, точное решение за время, близкое к линейному, и имеются соответствующие математические результаты.

Для краткости рассмотрим здесь только случай *одинаковых цен всех операций*. В [3] авторы описали сведение задачи с повторениями к задаче ЦЛП в случае, если структуры состоят только из циклических компонент. При этом число переменных и ограничений в соответствующей задаче ЦЛП не более чем квадратично от размера исходных структур, что, конечно, принципиально важно. Далее будет *описано такое сведение в общем случае с сохранением той же оценки на число переменных и число ограничений*.

3.2 Решение задачи

В исходных структурах a и b выберем произвольно второй индекс у всех повторяющихся имен; структуры с такими полными именами *обозначим* a' и b'; в них (полные) имена уже не повторяются.

Рассмотрим булевы переменные z_{abkij}, для которых $z_{abkij} = 1$, если ребро $k.i$ в a' по искомой биекции соответствует ребру $k.j$ в b', иначе $z_{abkij} = = 0$. Таким образом, значения этих переменных определяют соответствие ребер в a' и b'. Переименуем ребра в b' по этому соответствию, результат обозначим $a'(z)$ и $b'(z)$. С помощью ограничений

на эти переменные легко выразить понятия в $a'(z)$ и $b'(z)$: «особое ребро» и «цикл, состоящий из особых ребер» (назовем его *особым циклом*). Конечно, здесь описан только смысл переменной z, который выражен в рамках задачи ЦЛП.

Каждой паре s различных краев ребер в a' (или в b') сопоставим булеву переменную t_{as} (соответственно t_{bs}), ограничения на которые обеспечат следующие три свойства у $a'(z)$ (и у $b'(z)$): если край из s принадлежит общему ребру или лежит в особом цикле, то $t_{as} = 0$; для каждого края существует не более одного края, для которых $t_{as} = 1$ на этой паре в качестве s; для каждого края особого ребра, не принадлежащего особому циклу, существует край, для которого $t_{as} = 1$ на этой паре в качестве s. И аналогично для t_{bs}.

По значениям переменных t_{as} и t_{bs} определим новые вершины и ребра в $a'(z)$ и $b'(z)$, результат обозначим соответственно $a'(z,t)$ и $b'(z,t)$. Все особые ребра из $b'(z)$, не содержащиеся в особых циклах, добавим в $a'(z)$; края новых ребер склеим, если $t_{bs} = 1$ на этой паре в качестве s. Аналогично особые ребра из $a'(z)$, не содержащиеся в особых циклах, добавим в $b'(z)$; их края аналогично склеим, если $t_{as} = 1$. Таким образом, в структурах $a'(z,t)$ и $b'(z,t)$ все ребра общие, кроме принадлежащих особым циклам. Как раз эти особые циклы удалим из $a'(z,t)$ и $b'(z,t)$, результат обозначим теми же буквами; получены структуры с постоянным составом. Из условий на t_{as} и t_{bs} следует: каждое новое ребро входит в цикл из новых ребер.

Обозначим $G' = G'(z,t) = a'(z,t) + b'(z,t)$. Для графа G' вычислим значение $C_1 + 0{,}5C_2$, где C_1 и C_2 — число циклов и четных цепей в этом графе соответственно.

Число C_1 вычисляется так же, как в [3] для общего графа вычисляется число S_2 циклов, состоящих из обычных ребер (с учетом цепей и новых ребер).

Для вычисления величины $0{,}5C_2$ каждому краю p ребра в a и b сопоставим переменную r_p, принимающую значения 0, 1 или -1. Ограничения обеспечат условия: если для пары склеенных краев в $a'(z,t)$ или $b'(z,t)$ одна из переменных равна 1, то вторая равна -1, а если первая переменная равна 0, то вторая равна 0 или -1.

Переменные r_p будут входить в минимизируемую функцию F, которую определим чуть ниже, с отрицательным коэффициентом, поэтому они равны 1 на изолированных вершинах в G'. На вершинах из циклов в G' значения 1 и -1 переменных r_p чередуются или все эти значения нулевые; в любом случае сумма всех r_p вдоль цикла равна 0. Эти значения чередуются и на ненулевой четной цепи, причем на ее краях они равны 1; так что их сумма вдоль цепи равна 1. На нечетных цепях это чередование перемежается с нулевыми значениями; в любом случае их сумма вдоль такой цепи равна 0. Отсюда вытекает, что полусумма всех значений r_p равна C_2.

Итак, минимизируемая целевая функция *определяется как* $F = C_0 + n + s_a + s_b - C_1 - 0{,}5C_2$, где C_0 — сумма чисел особых циклов в $a'(z)$ и $b'(z)$; n — число (однократно учитываемых) общих ребер в них; s_a и s_b — число особых ребер в $a'(z)$ и $b'(z)$, не входящих в особые циклы. Значение $C_0 + n + s_a + s_b$ линейно выражается через введенные переменные z. Значение F линейно выражается через переменные z, t и r_p.

Итак, исходная задача сведена к задаче ЦЛП. Указанная оценка числа переменных и ограничений очевидна. Корректность такого сведения формально доказана в работе авторов, которая представлена в печать. В этом доказательстве решающий шаг состоит в том, что минимальное число операций в последовательности, преобразующей $a'(z)$ в $b'(z)$, равно F. Действительно, минимальная последовательность, преобразующая $a'(z,t)$ в $b'(z,t)$, имеет длину F, что следует из результата в [2]. Она индуцирует последовательность той же длины, преобразующую $a'(z)$ в $b'(z)$. И обратно: кратчайшая последовательность, преобразующая $a'(z)$ в $b'(z)$, индуцирует последовательность той же длины, преобразующую $a'(z,t)$ в $b'(z,t)$.

Идея такого соответствия последовательностей состоит в том, что операции удаления участка ребер ставится в соответствие стандартная операция, вырезающая и зацикливающая этот участок, а операции вставки участка ребер — стандартная операция, вставляющая этот зацикленный участок в то же место. Такая идея была предложена в [8].

Литература

1. *Горбунов К. Ю., Любецкий В. А.* Линейный алгоритм кратчайшей перестройки графов при разных ценах операций // Информационные процессы, 2016. Т. 16. № 2. С. 223–236.
2. *Горбунов К. Ю., Любецкий В. А.* Линейный алгоритм минимальной перестройки структур // Проблемы передачи информации, 2017 (в печати). Т. 53. Вып. 1.
3. *Lyubetsky V. A., Gershgorin R. A., Seliverstov A. V., Gorbunov K. Yu.* Algorithms for reconstruction of chromosomal structures // BMC Bioinformatics, 2016. Vol. 17. P. 40.1–40.23.
4. *Da Silva P. H., Machado R., Dantas S., and Braga M. D. V.* DCJ-indel and DCJ-substitution distances with distinct operation costs // Algorithm. Mol. Biol., 2013. Vol. 8. P. 21.1–21.15.

5. *Compeau P. E. C.* A generalized cost model for DCJ-indel sorting // Algorithms in bioinformatics / Eds. D. G. Brown, B. Morgenstern. — Lecture notes in computer science ser. — Springer, 2014. Vol. 8701. P. 38–51.
6. *Горбунов К. Ю., Гершгорин Р. А., Любецкий В. А.* Перестройка и реконструкция хромосомных структур // Молекулярная биология, 2015. Т. 49. № 3. С. 372–383.
7. *Горбунов К. Ю., Любецкий В. А.* Модифицированный алгоритм преобразования хромосомных структур: условия абсолютной точности // Современные информационные технологии и ИТ-образование, 2016. Т. 12. № 1. С. 162–172.
8. *Compeau P. E. C.* DCJ-indel sorting revisited // Algorithm. Mol. Biol., 2013. Vol. 8. P. 6.1–6.9.

Поступила в редакцию 26.04.16

HOW TO TRANSFORM A GRAPH INTO ANOTHER ONE WITH MINIMAL COST

K. Yu. Gorbunov[1] and V. A. Lyubetsky[1,2]

[1]A. A. Kharkevich Institute for Information Transmission Problems of the Russian Academy of Sciences, 19-1 Bolshoy Karetny Per., Moscow 127051, Russian Federation
[2]Faculty of Mechanics and Mathematics, M. V. Lomonosov Moscow State University, Main Building, Leninskiye Gory, GSP-1, Moscow 119991, Russian Federation

Abstract: The authors study orgraphs with any number of chains and cycles. Edges of orgraphs have unique names — natural numbers. There is a fixed list of operations that transform one graph into another. A cost is assigned to each operation. The task is to find the path of transformations with minimal total cost. This problem has a wide range of practical applications. The task is probably NP-hard and, thus, can be solved only under constraints imposed on costs or graphs. Such solutions are proposed in the study. The corresponding algorithms are linear in time and memory and are proved to be exact (nonheuristic), i. e., to find the path of transformations with minimal cost. Many heuristic algorithms solving this problem are known and tested on various data, but the proposed solutions are the first exact solutions.

Keywords: orgraph with chains and cycles; graph transformation; graph transformation with minimal total cost; exact linear algorithm; graph constraint; cost constraint; conditional shortest solution

DOI: 10.14357/19922264170107

Acknowledgments

The study was supported by the Russian Science Foundation (project No. 14-50-00150).

References

1. Gorbunov, K. Yu., and V. A. Lyubetsky. 2016. Lineynyy algoritm kratchayshey perestroyki grafov pri raznykh tsenakh operatsiy [A linear algorithm of the shortest transformation of graphs under different operation costs]. *Informatsionnye protsessy* [Information Processes] 16(2):223–236.
2. Gorbunov, K. Yu., and V. A. Lyubetsky. 2017 (in press). Lineynyy algoritm minimal'noy perestroyki struktur [Linear algorithm of the minimal reconstruction of structures under different operation costs]. *Problemy peredachi informatsii* [Problems of Information Transmission] 53(1).
3. Lyubetsky, V. A., R. A. Gershgorin, A. V. Seliverstov, and K. Yu. Gorbunov. 2016. Algorithms for reconstruction of chromosomal structures. *BMC Bioinformatics* 17:40.1–40.23.
4. Da Silva, P. H., R. Machado, S. Dantas, and M. D. V. Braga. 2013. DCJ-indel and DCJ-substitution distances with distinct operation costs. *Algorithm. Mol. Biol.* 8:21.1–21.15.
5. Compeau, P. E. C. 2014. A generalized cost model for DCJ-indel sorting. *Algorithms in bioinformatics*. Eds. D. G. Brown and B. Morgenstern. Lecture notes in computer science ser. Springer. 8701:38–51.
6. Gorbunov, K. Yu., R. A. Gershgorin, and V. A. Lyubetsky. 2015. Rearrangement and inference of chromosome structures. *Molecular Biology* 49(3):327–338.
7. Gorbunov, K. Yu., and V. A. Lyubetsky. 2016. Modifitsirovannyy algoritm preobrazovaniya khromosomnykh struktur: Usloviya absolyutnoy tochnosti [Modified algorithm for transformation of chromosome structures: Conditions of absolute accuracy]. *Sovremennye informatsionnye tekhnologii i IT-obrazovanie* [Modern Information Technologies and IT Education] 12(1):162–172.
8. Compeau, P. E. C. 2013. DCJ-indel sorting revisited. *Algorithm. Mol. Biol.* 8:6.1–6.9.

Received April 26, 2016

Contributors

Gorbunov Konstantin Yu. (b. 1965) — Candidate of Science (PhD) in physics and mathematics, leading scientist, A. A. Kharkevich Institute for Information Transmission Problems of the Russian Academy of Sciences, 19-1 Bolshoy Karetny Per., Moscow 127051, Russian Federation; gorbunov@iitp.ru

Lyubetsky Vassily A. (b. 1945) — Doctor of Science in physics and mathematics, professor, Head of Laboratory, A. A. Kharkevich Institute for Information Transmission Problems of the Russian Academy of Sciences, 19-1 Bolshoy Karetny Per., Moscow 127051, Russian Federation; professor, Faculty of Mechanics and Mathematics, M. V. Lomonosov Moscow State University, Main Building, Leninskiye Gory, GSP-1, Moscow 119991, Russian Federation; lyubetsk@iitp.ru