ДОКЛАДЫ РОССИЙСКОЙ АКАДЕМИИ НАУК. МАТЕМАТИКА, ИНФОРМАТИКА, ПРОЦЕССЫ УПРАВЛЕНИЯ, 2020, том 494, с. 26–29

МАТЕМАТИКА

УДК 519.178

ПОЧТИ ТОЧНЫЙ ЛИНЕЙНЫЙ АЛГОРИТМ ПРЕОБРАЗОВАНИЯ ГРАФОВ ИЗ ЦЕПЕЙ И ЦИКЛОВ, С ОПТИМИЗАЦИЕЙ СУММЫ ЦЕН ОПЕРАЦИЙ

© 2020 г. К. Ю. Горбунов[1,*], В. А. Любецкий[1,2,]**

Представлено академиком РАН А.Л. Семеновым 14.05.2020 г.
Поступило 17.05.2020 г.
После доработки 20.08.2020 г.
Принято к публикации 28.08.2020 г.

Построен алгоритм преобразования одного графа в другой для нагруженных ориентированных графов, составленных из цепей и циклов. Алгоритм работает линейное время и выдает последовательность преобразований с наименьшим, с точностью до аддитивной ошибки, суммарным весом, причем цены операций вставки и удаления участка ребер могут быть различными и отличаться от цены остальных операций. Аддитивная ошибка оценена через веса операций.

Ключевые слова: точный алгоритм, преобразование графов, граф степени 2, граф из цепей и циклов, цена операции, *DCJ*-операции, дискретная оптимизация

DOI: 10.31857/S2686954320050343

В работе описан алгоритм, который эффективно решает следующую задачу.

З а д а ч а. Д а н ы ориентированные графы *a* и *b*, у которых степень каждой вершины 1 или 2, и каждому ребру приписано имя – натуральное число. Вершина графа рассматривается как склеенная (отождествленная) из краев (концов) соседних ребер. Фиксированы 6 операций и каждой из них приписана цена – строго положительное число.

Н а й т и последовательность операций с наименьшей суммарной ценой, которая преобразует *a* в *b* (называемую кратчайшей последовательностью). Эти операции таковы:

1) *Rem*: удалить связный участок ребер с именами из *a*, но не из *b,* и наоборот,

2) *Ins*: вставить участок ребер с именами из *b*, но не из *a*; при удалении, если образовывались два свободных края (т.е. две вершины степени 1), они склеиваются, а при вставке сначала вершина расклеивается (если она не крайняя) и затем выполняются две склейки образовавшихся свободных краев.

Эти операции – обычные удаление подслова и вставка слова в качестве подслова. Далее,

3) *Cut*: расклеить какую-либо вершину или наоборот,

4) *OM*: склеить два свободных края;

в третьей операции из одной вершины степени 2 образуются две вершины степени 1, а в четвертой – из двух вершин степени 1 образуется одна вершина степени 2. И еще две операции, являющиеся композициями предыдущих:

5) *SM*: расклеить вершину и склеить один из образовавшихся свободных краев с каким-либо уже свободным краем;

6) *DM*: расклеить две вершины и склеить образовавшиеся свободные края. Последние четыре операции вместе называются *DCJ*, они определены в [1].

Это заканчивает постановку задачи.

Обзор предшествующих результатов по этой задаче, включая ее прикладные аспекты, содержится в сборнике [2, Chap. 10]. Во всех предшествующих работах других авторов, из которых сошлемся на последнюю по времени работу [4], задача решалась при условии равных цен *DCJ*-операций и равных цен операций *Rem* и *Ins*, что принципиально упрощает задачу по сравнению с условиями в теореме 1. Теорема 1 утверждает линейность по времени работы и аддитивную точность Алгоритма, который описан после нее.

[1] *Институт проблем передачи информации им. А.А. Харкевича Российской академии наук, Москва, Россия*

[2] *Московский государственный университет имени М.В. Ломоносова, Москва, Россия*

**E-mail: gorbunov@iitp.ru*

***E-mail: lyubetsk@iitp.ru*

Обозначим цены операций *Rem* и *Ins* соответственно w_d и w_i.

Теорема 1. *Если DCJ-операциям приписаны равные цены w, то Алгоритм выдает последовательность со значением суммарной цены, отличающимся от ее минимального значения не больше чем на аддитивную константу k, зависящую только от цен операций. А именно, если* $\max\{w_d, w_i\} \le w$, *то* $k = 0$; *если* $\min\{w_d, w_i\} \ge w$, *то* $k = 2w$; *если* $\min\{w_d, w_i\} < w < \max\{w_d, w_i\}$, *то* $k = w_i - 1$ (*при выполнении* $w_d + w_i \le 2w$), $k = 4w_i + 2w_d - 6$ (*при* $w_d + w_i > 2w$ *и* $\max\{w_d, w_i\} \le 2w$) *и* $k = 6w_i + 2w_d - 9$ (*при* $\max\{w_d, w_i\} > 2w$).

Из доказательства теоремы 1 следует, что на самом деле k (если не нуль) можно сделать заметно меньше, но мы не выписываем соответствующие более сложные выражения.

Оставшаяся часть сообщения содержит описание Алгоритма и его свойств. Начнем со вспомогательных определений.

Нам понадобится определение графа $a + b$ из [5], а в иных вариантах оно известно из более ранних работ, например, [6]. Напомним

Определение [5]. Вершины в $a + b$ – все края ребер, присутствующих одновременно в a и b; и еще вершины, однозначно сопоставленные каждому максимальному связному участку ребер ("блоку") в $a\backslash b$ или в $b\backslash a$, которые помечены соответственно a или b;

ребра в $a + b$ соединяют вершины, если таковые склеены в a или в b или край блока склеен с вершиной в a или в b, ребра помечены соответственно a или b.

В $a + b$ сингулярной назовем вершину внутри пары ребер aa или bb, и также – помеченный край цепи или помеченную изолированную вершину; остальные вершины назовем обычными. Ребро, край которого сингулярный, назовем сингулярным; другие ребра – обычными.

Финальный вид – граф, состоящий из циклов длины 2, у каждого цикла одно ребро помечено a и другое b, и обычных изолированных вершин. В $a + b$ цена удаления становится ценой w_a удаления a-сингулярной вершины, а цена вставки – ценой w_b удаления b-сингулярной вершины. *DCJ*-операции остаются теми же, только вместо удаления участка используется удаление a-сингулярной вершины, а вместо вставки участка – удаление b-сингулярной вершины.

Висячим назовем ребро, инцидентное сингулярной вершине степени 1. Размером компоненты в графе $a + b$ назовем число обычных ребер плюс половина числа ее сингулярных невисячих ребер; размер равен 0 для обычных изолированных вершин и петель, и равен –1 для сингулярных изолированных вершин. Цепь нечетного (четного) размера назовем нечетной (четной). Пусть цепь не содержит обычных ребер; определим типы таких цепей. Тип 1a – нечетная цепь с одним висячим b-ребром; тип 2a* – нечетная цепь с двумя висячими b-ребрами; тип 2a' – b-сингулярная изолированная вершина; тип 2a – тип 2a* или 2a'. Тип 3a* – нечетная цепь без висячих ребер с двумя крайними a-ребрами, имеющая b-сингулярную вершину; тип 3a' – цепь aa; тип 3a –тип 3a* или 3a'. Тип 1_a^* – четная цепь с одним висячим a-ребром, имеющая b-сингулярную вершину; тип $1_a'$ – висячее a-ребро; тип 1_a – тип 1_a^* или $1_a'$. Аналогично с заменой a на b. Тип 2* – четная цепь с двумя висячими не инцидентными друг другу ребрами; тип 2' – два висячих ребра, инцидентных общей обычной вершине; тип 2 – тип 2* или 2'. Тип 3 – четная цепь без висячих ребер, но с сингулярными вершинами. Тип 0 – цепь без сингулярных вершин. Тип цепи с обычными ребрами определим как тип цепи, полученной после их удаления, такое определение не зависит от того, в каком порядке оно выполнялось [5, лемма 6]. (a, b)-Цикл – цикл с сингулярными a- и b-вершинами; a-цикл – цикл с сингулярными a-вершинами, но без сингулярных b-вершин; аналогично определяется b-цикл.

Алгоритм начинает работу на графе $a + b$, последовательно порождая графы G вплоть до графа финального вида; все G из такой последовательности имеют вид $c + d$ для своих графов c и d. Последовательность, которая начинается с $a + b$ и заканчивается графом финального вида, называется приводящей. Если для $a + b$ найдена кратчайшая приводящая последовательность, то по ней за линейное время строится искомое кратчайшее преобразование исходных графов a в b.

Алгоритм состоит из 8 этапов.

Этап 0: преобразовать исходную пару графов a и b в новый ("breakpoint") граф $a + b$. При этом исходная задача эквивалентна задаче приведения этого графа $a + b$ к финальному виду аналогами исходных операций над a и b. Доказательство эквивалентности этих двух задач дословно повторяет доказательство следствия 5 из [5], в котором используется только равенство цен *DCJ*-операций.

Этап 1: из всех компонент нефинального вида удалим обычные ребра, т.е. применим *DM* к паре соседних ребер или *SM*, если отсутствует одно из соседних ребер, или *OM*, если отсутствуют оба соседних ребра.

Этап 2: выполним найденные нами композиции исходных операций, которые назовем взаимодействиями. Эти композиции применяются к цепям, тип которых указан в левой части одного из следующих равенств, при этом тип результата – тот, который указан в правой части равенства. Итак, 2-взаимодействия – это *SM* со следующи-

ми равенствами термов (разрезаемая цепь везде указана первой, обычные изолированные вершины не представлены в правой части):

$1a + 1b = 1^*_b$, $3a + 2b = 1_a$, $3b + 2a = 1_b$, $3 + 2 = 1^*_b$, $(1a + 2b) + 3 = 1^*_b$, $(1b + 2a) + 3 = 1^*_b$, $(3a + 1b) + 2 = 1^*_b$, $(3b + 1a) + 2 = 1_b{}^*$, $1a + 2 = 2a^*$, $1b + 2 = 2b^*$, $3 + 1a = 3a^*$, $3 + 1b = 3b^*$, $(3b + 1a) + (1a + 2b) = 1^*_b$, $(3a + 1b) + (1b + 2a) = 1^*_b$, $1a + (1a + 2b) = 2a^*$, $1b + (1b + 2a) = 2b^*$, $(3b + 1a) + 1a = 3a^*$, $(3a + 1b) + 1b = 3b^*$, $1a + 2b = 2$, $1b + 2a = 2$, $3a + 1b = 3$, $3b + 1a = 3$, $3 + ((3 + 2b) + 2a) = 1^*_b$, $((3a + (3b + 2)) + 2 = 1^*_b$, $(3a + 2) + 2 = 2a^*$, $(3b + 2) + 2 = 2b^*$, $3 + (3 + 2a) = 3a^*$, $3 + (3 + 2b) = 3b^*$, $(3 + 2b) + 2a = 2^*$, $3a + (3b + 2) = 3$ и *OM* с равенствами $1a + 1a = 3a^*$ или $1b + 1b = 3b^*$. Если $\min\{w_d, w_i\} < w$, то выполним еще два 2-взаимодействия, определяемые как *SM*: $3a^* + 3b = 3$ или $2a + 2b^* = 2^* + 1'_a$.

Каждое 2-взаимодействие применяется, пока это возможно, если $\min\{w_d, w_i\} \geq w$; иначе строится максимальный набор взаимодействий с непересекающимися аргументами.

Эти и нижеуказанные взаимодействия строго уменьшают сложность текущего графа G, которая измеряется величиной $t(G)$ – минимальным числом операций в приведении G. Принципиально, что $t(G)$ выражается через простые характеристики G (кроме одной характеристики, обозначаемой $P(G)$, которую, однако, удается вычислить, см. ниже), и потому в целом $t(G)$ легко вычисляется. Это – основная идея как алгоритма, так и доказательства его точности.

На этапах 3, 5–7 ниже каждое взаимодействие применяется, пока это возможно.

Э т а п 3: если $\max\{w_d, w_i\} > w$, то выполним следующие *3*-взаимодействия, которые уменьшают число компонент в графе G и также уменьшают его сложность: *DM* (но *SM* для изолированной вершины): *b*-петля + "компонента с *b*-сингулярной вершиной" = та же компонента; *SM*: $2 + 2 = 2 + 1'_a$ (*SM* отрезает висячее *a*-ребро и склеивает образовавшийся край с крайней *b*-сингулярной вершиной другой цепи); *OM*: $2a + 2a = 2a$, $2b^* + 2b^* = 2b^*$ (склейка висячих вершин двух цепей); *OM* и *DM*:

$3a^* + 3a^* = 3a^*$, $3b + 3b = 3b$ (после *OM* еще удаление обычного ребра); *SM*: $3a^* + 2 = 1a$, $3b + 2 = 1b$; $3 + 2a = 1a$, $3 + 2b^* = 1b$; *OM* и *DM*: $1a + 3a^* = 1a$, $1b + 3b = 1b$ (как выше); *OM*: $1a + 2a = 1a$, $1b + 2b^* = 1b$; *OM* и *DM*: $3a^* + 3 = 3$, $3b + 3 = 3$ (как выше); *OM*: $2a + 2 = 2$, $2b^* + 2 = 2^*$; SM: $3 + 3 = 3$; $1^*_a + 1^*_a = 1^*_a$, $1_b + 1_b = 1_b$; $1_b + 1a = 1a$, $1^*_a + 1b = 1b$; $1a + 1^*_a = 1a$, $1b + 1_b = 1b$; $1_b + 2a = 2a$, $1^*_a + 2b^* = 2b^*$; $3a^* + 1^*_a = 3a^*$, $3b + 1_b = 3b$; $3 + 1^*_a = 3$, $3 + 1_b = 3$; $1^*_a + 2 = 2^*$, $1_b + 2 = 2$.

3-Взаимодействия показаны на рисунках в [3, с. 8–9].

Э т а п 4: цепи строго положительного размера замкнем в циклы операциями *OM* (если цепь нечетная) или *SM* (если четная; остается крайняя изолированная обычная вершина или крайнее изолированное висячее ребро).

Э т а п 5: если $w_i + w_d > 2w$, выполним следующие *5*-взаимодействия. Дважды *DM*: (a, b)-цикл + (a, b)-цикл = (a, b)-цикл = (a, b)-цикл + цикл длины 2 (вторая *DM* удаляет обычное ребро); дважды *DM*: (a, b)-цикл размера строго больше двух = (a, b)-цикл того же размера с обычным ребром = (a, b)-цикл меньшего размера + цикл длины 2 (удаление обычного ребра). Они показаны на рисунках в [3, с. 9]. Если $w_i + w_d \leq 2w$, операцией *DM* от каждого цикла размера строго больше 2 отщепим цикл размера 2 с одной сингулярной *a*-вершиной (если $w_i \geq w_d$) или с *b*-вершиной (если $w_i < w_d$).

Э т а п 6: если $w_i + w_d > 2w$, выполним *6*-взаимодействия между (a, b)-циклом размера 2 и цепями размера 0. Дважды *SM*: (a,b)-цикл + $2a' = 1b$ и затем $1b + 2b' = 2'$. Дважды *DM*: (a,b)-цикл + $2' = 2'$ с обычным ребром = $2'$ + цикл длины 2 (удаление обычного ребра). Они показаны на рисунках в [3, с. 9–10].

Э т а п 7: если $\max\{w_d, w_i\} > 2w$, выполним следующие *7*-взаимодействия с циклами размера 2 и цепями размера 0. Дважды *DM*: (a, b)-цикл + + *a*-цикл = (a, b)-цикл размера 4 с двумя обычными ребрами = (a, b)-цикл + цикл длины 2 (удаление обычного ребра). Дважды *DM*: *a*-цикл + + *a*-цикл = *a*-цикл размера 4 с тремя обычными ребрами = *a*-цикл + цикл длины 2 (удаление обычного ребра). Дважды *SM*: *a*-цикл + $2' = 2'$ с одним обычным крайним ребром = $2'$ + цикл длины 2 (удаление обычного ребра). *SM* и *OM*: (a, b)-цикл + + $2b' = 1b = (a, b)$-цикл. Дважды *SM*: (a, b)-цикл + $1'_a = 3 = (a, b)$-цикл. Дважды *SM*: *a*-цикл + $2b' = 2b'$ с одним обычным крайним ребром = $2b'$ + цикл длины 2 (удаление обычного ребра). Дважды *SM*: *a*-цикл + $1'_a = 1'_a$ с одним обычным крайним ребром = $1'_a$ + цикл длины 2 (удаление обычного ребра). *7*-Взаимодействия показаны на рисунках в [3, с. 10]. Они заменяют удаление сингулярной вершины на две *DCJ*-операции, что выгодно при $\max\{w_d, w_i\} > 2w$. Эти взаимодействия применим для случая $w_d > 2w$. Если $w_i > 2w$, применим их с переменой местами *a* и *b*; если оба неравенства, то применим те и другие.

Э т а п 8: удалим сингулярные вершины.

Линейное время работы алгоритма следует из того, что граф $a + b$ строится однократным просмотром компонент в *a* и *b*. Также этап 1 алгоритма требует однократного просмотра компонент в $a + b$. Число выполняемых в алгоритме операций

линейно, так как каждое взаимодействие, состоящее не более чем из трех операций, уменьшает число вершин в $a + b$ и не увеличивает число ребер в нем, или уменьшает число ребер в нефинальной части $a + b$ и не увеличивает число его вершин. А каждая операция выполняется за константное время.

Пусть $T(G)$ – суммарная цена операций в последовательности, которую выдает алгоритм на G. Доказательство точности в теореме 1 основано на неравенствах типа: для любой исходной операции o и любого графа G выполняется $c(o) \geq T(G) - T(o(G))$, где $c(o)$ – цена операции o и $o(G)$ – результат применения o к G.

ИСТОЧНИК ФИНАНСИРОВАНИЯ

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта 18–29–13037.

СПИСОК ЛИТЕРАТУРЫ

1. *Yancopoulos S., Attie O., Friedberg R.* Efficient Sorting of Genomic Permutations by Translocation, Inversion and Block Interchange // Bioinformatics. 2005. № 21. P. 3340–3346. https://doi.org/10.1093/bioinformatics/bti535
2. Bioinformatics and Phylogenetics. Seminal Contributions of Bernard Moret. Eds. *Tandy Warnow*. Comput. Biol. Series. Springer Nature Switzerland AG, 2019.
3. *Gorbunov K.Yu., Lyubetsky V.A.* An Almost Exact Linear Complexity Algorithm of the Shortest Transformation of Chain-Cycle Graphs // arXiv:2004.14351 [math.CO]. 2020.
4. *da Silva P.H., Machado R., Dantas S., Braga M.D.V.* Genomic Distance with High Indel Costs // J. IEEE/ACM Trans. Comput. Biol. Bioinform. 2017. V. 14. № 3. P. 1–6.
5. *Горбунов К.Ю., Любецкий В.А.* Линейный алгоритм минимальной перестройки структур // Проблемы передачи информации. 2017. Т. 53. Вып. 1. С. 60–78.
6. *Alekseyev M.A., Pevzner P.A.* Multi-Break Rearrangements and Chromosomal Evolution // Theor. Computer Science. 2008. V. 395. № 2–3. P. 193–202. https://doi.org/10.1016/j.tcs.2008.01.013

AN ALMOST EXACT LINEAR ALGORITHM OF THE TRANSFORMATION OF CHAIN-CYCLE GRAPHS WITH OPTIMIZATION OF THE SUM OF OPERATION COSTS

K. Yu. Gorbunov[a] and V. A. Lyubetsky[b]

[a] *Institute for Information Transmission Problems of the Russian Academy of Sciences (Kharkevich Institute), Moscow, Russian Federation*

[b] *Lomonosov Moscow State University, Moscow, Russian Federation*

Presented by Academician of the RAS A.L. Semenov

The authors report the proof of the originally proposed linear by time algorithm to almost exactly solve the discrete optimization problem of the shortest transformation between any two chain-cycle graphs with predefined operation costs under the equalness costs of *DCJ*-operations.

Keywords: exact algorithm, graph transformation, 2-degree graph, chain-cycle graph, operation cost, total cost minimization