

OPTIMIZATION OF THE k -SHORTEST PATHS ALGORITHM

A. R. Gerb, E. E. Deviatykh, G. A. Omarova

Institute of Computational Mathematics and Mathematical Geophysics SB RAS,
630090, Novosibirsk, Russia
Novosibirsk State University
630090, Novosibirsk, Russia

DOI: 10.24412/2073-0667-2026-1-40-49

EDN: RGXRWO

This paper examines algorithms for finding k -shortest paths. An optimized algorithm for finding k -shortest simple (loop-free) paths in a directed graph is implemented. The algorithm is based on the ideas of the classical version of Yen's algorithm and the reverse tree construction algorithm. Both implementations have complexity $O(kn(m + n \log n))$.

Key words: graph, path, simple path, tree, graph algorithm, k shortest simple paths, space-time trade-off.

References

1. Egli W., Kraus M. Shortest paths in chemical kinetic applications // Phys. Chem. Chem. Phys. 2003. Vol. 5. P. 3916–3920.
2. Yen Jin Y., Finding the K shortest loopless paths in a network // Manag. Sci. 1971. Vol. 17. P. 712–716.
3. Hoffman R., Pavley R. A method for the solution of the n th best path problem // J. of the Association for Computing Machinery. 1959. Vol. 6, N 4. P. 506–515.
4. Bellman R., Kalaba R. On k th best policies // J. of SIAM. 1960. Vol. 8, N 4. P. 582–588.
6. Bock F., Kanther H., Haynes J. An algorithm (the rh best path algorithm) for finding and ranking paths through a network // Research Report, Armour Research Foundation. Chicago, Illinois, 1957.
7. Clarke S., Krikorian A., Rausan J. Computing the N best loopless paths in a network // J. of SIAM. 1963. Vol. 11, N 4. P. 1096–1102.
9. Dijkstra E. W. A note on two problems in connexion with graphs // Numerische Mathematik. 1959. Vol. 1. P. 269–271.
8. Pollack M. The k th best route through a network // Operations Research. 1961. Vol. 9, N 4. P. 578.
5. Sakarovitch M. The k shortest routes and the k shortest chains in a graph // Operations Research Center Report N 66–32. University of California, Berkeley. 1966.
10. Coudert D., Al Zoobi A., Nisse N. Space and time trade-off for the k shortest simple paths problem space and time trade-off for the k shortest simple paths problem // 18th International Symposium on Experimental Algorithms (SEA 2020). Art. N 18. P. 18:1–18:13.

This work was carried out under state contract with ICMMG SB RAS FWNM-2025-0001.

11. Kurz D., Mutzel P. A sidetrack-based algorithm for finding the k shortest simple paths in a directed graph // In Int. Symp. on Algorithms and Computation (ISAAC), Schloss Dagstuhl, 2016. Vol. 64 of LIPIcs. P. 49:1–49:13.
12. Frigioni D., Marchetti-Spaccamela A., Nanni U. Fully dynamic algorithms for maintaining shortest paths trees // J. of Algorithms. 2000. Vol. 34, N 2. P. 251–281.
13. [Electrol. Res.]: <https://www.diag.uniroma1.it/challenge9/download.html>.
14. [Electrol. Res.]: <https://snap.stanford.edu/data/ca-AstroPh.html>.
15. [Electrol. Res.]: <https://snap.stanford.edu/data/ca-CondMat.html>.
16. [Electrol. Res.]: <https://snap.stanford.edu/data/ca-GrQc.html>.
17. [Electrol. Res.]: <https://snap.stanford.edu/data/ca-HepTh.html>.

ОПТИМИЗАЦИИ АЛГОРИТМА ПОИСКА k КРАТЧАЙШИХ ПУТЕЙ

А. Р. Герб, Е. Е. Девярых, Г. А. Омарова

Институт вычислительной математики и математической геофизики СО РАН,
630090, Новосибирск, Россия
Новосибирский государственный университет
630090, Новосибирск, Россия

УДК 519.17+51-7

DOI: 10.24412/2073-0667-2026-1-40-49

EDN: RGXRWO

В работе исследуются алгоритмы поиска k кратчайших путей. Реализован оптимизированный алгоритм для нахождения k кратчайших простых (без петель) путей в ориентированном графе. Алгоритм основан на идеях классической версии алгоритма Йена и алгоритма построения обратного дерева, обе реализации имеют сложность $O(kn(m + n \log n))$.

Ключевые слова: граф, путь, простой путь, дерево, обход графа.

Введение. Практические задачи вычисления кратчайшего пути слишком многочисленны. Они включают ситуации, в которых фактический путь является желаемым выходом, планирование движения робота, проектирование автомагистралей и линий электропередач, а также маршрутизации сетевых соединений. К ним также относятся задачи планирования, вычисление критического пути в диаграммах. Задачи оптимизации, решаемые методом динамического программирования или более сложными методами поиска матриц, выравнивание последовательностей в молекулярной биологии, построение оптимальных вписанных многоугольников могут быть выражены как задачи поиска кратчайшего пути.

Задача о k кратчайших путях является естественным и давно изученным обобщением задачи о кратчайших путях, в которой ищется не один, а несколько путей в порядке возрастания длины. Для заданного ориентированного графа G с неотрицательными весами ребер, положительного целого числа k и двух вершин s и t требуется найти k кратчайших путей из s в t в порядке возрастания длины. Методы поиска k кратчайших путей используются во многих задачах. Иногда возникает необходимость найти путь, удовлетворяющий определенным ограничениям, помимо небольшой длины, однако эти ограничения могут быть плохо определены или их трудно оптимизировать. Типичное решение — вычислить несколько коротких путей, и затем выбрать один из них с учетом других критериев. Пути могут использоваться для моделирования задач, имеющих известные решения, независимо от формулировки пути. Например, в модели k кратчайших путей автоматического перевода между естественными языками правильный перевод может быть найден экспертом-человеком. Перечисляя пути до появления этого известного решения, можно определить,

Исследования выполнены в рамках государственного задания ИВМиМГ СО РАН FWNM-2025-0001.

насколько хорошо модель соответствует задаче с точки зрения количества неверных путей, обнаруженных до правильного. Информацию можно использовать для настройки модели, а также для определения количества путей, которые необходимо сгенерировать при применении дополнительных ограничений для поиска правильного решения.

Вычисляя более одного кратчайшего пути, можно определить, насколько чувствительно оптимальное решение к изменению параметров задачи. Полезно изучить не только оптимальное решение задачи, но и более широкий класс решений, чтобы лучше понять проблему. Например, состояния сложной системы можно представить как конечный автомат, по сути просто граф с различными вероятностями на каждом ребре перехода состояний. В такой модели, вероятно, потребуется знать не только цепочку событий, которая с наибольшей вероятностью приведет к состоянию отказа, но и все цепочки, вероятность отказа которых превышает некоторый порог. Логарифмирование вероятностей перехода превращает эту задачу в задачу поиска всех путей короче заданной длины.

Применение концепции кратчайшего пути и алгоритма поиска k кратчайших путей в химическом анализе было предложено Эгли в 2003 году [1]. В работе был построен оргграф, узлы которого включали не только вещества, но и реакции. Хотя работа не была направлена на создание метода генерации скелетных механизмов, она предложила новую и полезную идею выделения основных путей реакции.

В данной работе представлена реализация алгоритма ранжирования беспетлевых путей Йена с оптимизацией. Обычная реализация прямо следует классической версии алгоритма Йена [2], оптимизация использует подход построения обратного дерева, поскольку узлы каждого k -го кратчайшего пути без петель анализируются в обратном порядке. Хотя обе реализации имеют сложность $O(kn(m + n \log n))$, в худшем случае анализ новой реализации показал лучшую производительность в вычислительных экспериментах, проведенных для оценки их поведения в среднем случае.

Решение задачи поиска k кратчайших путей важно для ряда задач оптимизации и часто используется для ранжирования путей по неубывающему порядку их стоимости, пока не будут найдены один или несколько путей с небольшой стоимостью, удовлетворяющих некоторым ограничениям.

1. Алгоритм поиска k кратчайших путей. Одно из первых решений задачи поиска k кратчайших путей, или ранжирования k кратчайших путей между парой узлов в сети, было предложено Хоффманом и Павли в 1959 г. [3]. Авторы описали алгоритм, в котором для заданного целого числа $k \geq 1$ последовательно определяются 1-й кратчайший путь, 2-й кратчайший путь и так до k -го кратчайшего пути между заданной парой узлов. Существуют два типа задач поиска k кратчайших путей. Первый тип – это нахождение путей от источника s до стока t , имеющих наименьшую длину, в которых петли разрешены. Алгоритмы для решения этого типа задач предложены в работах [3, 4, 5]. Второй тип задач – поиск k путей от s до t , имеющих наименьшую длину, в которых петли не допускаются [6, 7, 8, 5].

В работе [6] авторы вводят перечисление всех возможных путей из k кратчайших путей в сети. Процедура перечисляет все пути от s до t , а затем выбирает из них k путей с наименьшей длиной. Данный алгоритм имеет два основных недостатка: во-первых, требуется очень большое количество вычислений и памяти; во-вторых, для решения задачи с малым k требуется столько же усилий, сколько и для решения задачи с большим k , и чем больше k , тем затруднительнее использование этого алгоритма.

В работе [8] автор для поиска k -го кратчайшего пути сначала получает $k - 1$ кратчайший путь. Затем расстояние каждой дуги в каждом из 1-го, 2-го, ..., $(k - 1)$ -го кратчайших путей поочередно устремляется к бесконечности. Далее задача поиска кратчайшего пути решается для каждого такого случая. Лучший из этих полученных кратчайших путей является искомым k -м кратчайшим путем. Количество вычислений, требуемых для этой процедуры, экспоненциально возрастает с ростом значения k , следовательно, если k не мало, алгоритм Поллака [8] перегружает вычислительные ресурсы.

В работе [7] авторы предлагают процедуру ветвей и границ для поиска k кратчайших путей в сети: сначала находится кратчайший путь, а затем — k кратчайших путей среди всех путей, ответвляющихся от кратчайшего. Эффективность этого алгоритма зависит от конкретной сети. Если второй кратчайший путь «немедленно ответвляется» от первого, третий кратчайший путь «немедленно ответвляется» от второго кратчайшего пути и т. д., эта процедура может очень быстро определить все k кратчайших путей. Однако в общем случае этот алгоритм, вероятно, потребует огромного количества вычислений и памяти. Поэтому эффективность этой процедуры сложно оценить.

В работе [5] автор описывает следующий алгоритм поиска k кратчайших путей: сначала с помощью процедуры, аналогичной (но менее эффективной), чем алгоритм Хоффмана и Павли, находятся $H > k$ кратчайших путей, которые могут содержать петли [3]. Затем H путей исследуются на k кратчайших путей, не содержащих петель. Эффективность этого алгоритма зависит от конкретной сети. Если H кратчайших путей, полученных алгоритмом Хоффмана, не содержат петель, алгоритм Сакаровича [5] может очень быстро определить k кратчайших путей без петель. Однако для этого алгоритма сложно указать верхнюю вычислительную границу.

1.1. Определения. Пусть $G = (V, E)$ — ориентированный граф с множеством вершин V и множеством ребер E ; $n = |V|$ — количество вершин; $m = |E|$ — количество ребер в графе G .

Подграфом в графе $G(V, E)$ называется граф $H(V_H, E_H)$, такой что $\forall v \in V_H, v \in V$ и $\forall e \in E_H, e \in E$.

Пусть $e \in E$, такое что $e = (u, v)$, тогда обозначим $head(e) = v$ — начало дуги e и $tail(e) = u$ — конец дуги e .

Обозначим $w_G : E \rightarrow R^+$ функцию веса для ребер графа G .

Пусть $u, v \in V$, тогда путем из вершины u в вершину v в графе G называется последовательность вершин $P = P_{uv}^G = (v_0 = u, v_1, \dots, v_r = v)$, такая что $(v_i, v_{i+1}) \in E$ для $\forall i : 0 \leq i < r$. Путь P будем обозначать как $u \rightarrow v$.

Путь называется простым тогда и только тогда, когда $v_i \neq v_j$ для $\forall i, j : 0 \leq i < j \leq r$.

Для возможности вычисления длины пути расширим функцию w_G на множество всех путей следующим образом: $w_G(P) = \sum_{i=0}^{r-1} w_G(v_i v_{i+1})$.

Обозначим путь $v_1 \rightarrow w_p$ как $(v_1, \dots, v_r, Q)$ — путь, полученный сложением двух путей $(v_1, \dots, v_r)$ и $Q = (w_1, \dots, w_p)$, если $(v_r, w_1) \in E$.

Пусть существует путь $u \rightarrow v$, тогда длину кратчайшего пути $u \rightarrow v$ в графе G обозначим $d_G(u, v)$.

Пусть $t \in V$, тогда обратным деревом кратчайших путей T с корнем в t назовем любой подграф G , для которого выполняются условия: 1) $\forall u \in T \setminus \{t\}$ имеет максимум одно исходящее ребро, а t вообще не имеет таких ребер; 2) $\forall u \in T$ длина пути P_{ut}^T равна $d_G(u, t)$, то есть P_{ut}^T является кратчайшим путем.

Пусть $s, t \in V$, тогда S — множество k попарно неравных кратчайших путей, такое что $w_G(P_1) \leq w_G(P_2)$, где $P_1 \in S$ и $P_2 \notin S$.

1.2. *Описание алгоритма Йена поиска k кратчайших путей.* Задача поиска k кратчайших путей заключается в нахождении k путей, имеющих наименьшую длину от начальной до конечной вершины взвешенного графа, при этом каждый путь является простым, то есть не посещает одну и ту же вершину более одного раза [2]. Алгоритм Йена является классическим методом решения задачи в ориентированных графах с неотрицательными весами дуг.

Алгоритм строится на основе индукции по мощности множества кратчайших путей и работает до тех пор, пока не будет найдено заданное количество кратчайших путей.

База индукции $k' = 1$. Вычисляется кратчайший путь $P_0 = (v_0 = s, v_1, \dots, v_r = t)$ от вершины s до вершины t с помощью алгоритма Дейкстры [9] (P_0 существует, иначе решений нет). Второй кратчайший путь является простым кратчайшим ответвлением P_0 .

Алгоритм Йена вычисляет ответвление для каждой вершины v_i пути P_0 следующим образом:

1) Для $\forall i, i = \overline{0, r-1}$ создается подграф $G_i(P_0)$ графа G путем удаления вершин $v_0, \dots, v_{i-1}$, чтобы избежать ответвления, не являющегося простым, и удалением ребра (v_i, v_{i+1}) , чтобы получить путь, отличающийся от P_0 .

2) Для $\forall i, i = \overline{0, r-1}$ вычисляется кратчайший путь Q_i от v_i до t в подграфе $G_i(P_0)$ с использованием алгоритма Дейкстры. Если не существует пути, то данное i не рассматривается на шаге 3.

3) Для $\forall i, i = \overline{0, r-1}$ простое ответвление $(v_0, \dots, v_i, Q_i)$ пути P_0 добавляется в множество кандидатов C . Индекс ответвления i сохраняется явно в паре с ответвлением $(v_0, \dots, v_i, Q_i)$ для последующего использования.

Вторым искомым путем будет самый короткий путь из множества кандидатов C .

Индукционное предположение для $0 < k' < k$. Предположим, что множество k' кратчайших путей S вычислено, множество кандидатов C содержит простые пути $s \rightarrow t$, а также существует кратчайший путь $Q \in C$, такой что $S \cup Q$ является множеством $k' + 1$ кратчайших путей.

Шаг индукции $k' \rightarrow k' + 1$. Пусть $Q = (v_0 = s, \dots, v_r = t)$ и $j (0 \leq j < r)$ — индекс ответвления. Q удаляется из множества кандидатов C и добавляется в результирующее множество S . Затем вычисляются все простые ответвления и добавляются в множество кандидатов C :

1) Для $\forall j, j \leq i < r$ создается подграф $G_i(Q)$ графа G путем удаления вершин $v_0, \dots, v_{i-1}$ и всех дуг вида (v_i, w) , таких что множество кандидатов C уже содержит путь с префиксом $(v_0, \dots, v_i, w)$.

2) Для $\forall i, j \leq i < r$ вычисляется кратчайший путь Q_i от v_i до t в $G_i(Q)$. Если не существует пути, то данное i не рассматривается на шаге 3.

3) Для $\forall j, j \leq i < r$ ответвление $(v_0, \dots, v_i, Q_i)$ пути Q добавляется в множество кандидатов C с его индексом ответвления i .

Следующим $(k' + 2)$ кратчайшим путем будет кратчайший путь из множества кандидатов.

1.3. *Оптимизации алгоритма поиска.* Оптимизация алгоритма проведена на основе алгоритма, предложенного в работе Али Аль Зуби [10], улучшающего метод поиска k кратчайших путей Даниэля Курца и Петра Муцеля [11]. Улучшение направлено на уменьшение потребления оперативной памяти.

Основная идея алгоритма сводится к построению обратных деревьев кратчайших путей. Пусть C_{simple} – приоритетная очередь, построенная на основе двоичной кучи. Данная очередь хранит первые k простых путей кандидатов и их длину в виде $((T_0, e_0, T_1, \dots, T_h, e_h, T_{h+1}), lb)$, где T_i – обратное дерево кратчайших путей; $e_i \in E$ – дуга; lb – длина пути в графе G . Дерево T_{h+1} может храниться в множестве T_{cache} , в противном случае оно вычисляется с помощью алгоритма Дейкстры. При этом $\forall i, i = \overline{0, h}$ T_i гарантированно вычислено и хранится в T_{cache} . Чтобы извлечь из данного представления путь, необходимо пройти от вершины s до $tail(e_0)$ в обратном дереве кратчайших путей T_0 , проследовать по дуге e_0 в T_1 до $tail(e_1)$ и так далее до тех пор, пока не будет достигнута вершина t дерева T_{h+1} .

Вторая очередь $C_{not-simple}$ хранит группы ответвлений-кандидатов с их нижней оценкой длин lb . Данные группы кандидатов требуют явного вызова алгоритма Дейкстры для решения проблемы их простоты.

В начале работы алгоритма вычисляется обратное дерево кратчайших путей T_0 , содержащее кратчайший путь $s \rightarrow t$, если такого пути нет, то нет и решения задачи. Дерево T_0 добавляется в C_{simple} . Цикл выполняется до тех пор, пока не будет найдено k кратчайших путей или не исчерпаны очереди C_{simple} и $C_{not-simple}$. В последнем случае результирующее множество S будет содержать меньше k элементов.

В каждой итерации цикла алгоритм сначала выбирает лучший элемент φ из двух очередей (с приоритетом для C_{simple} , если оценки lb для лучших элементов обеих очередей совпадают). Если φ был извлечен из C_{simple} , то φ имеет вид $((T_0, e_0, T_1, \dots, T_h, e_h = (u_h, v_h), T_{h+1}), lb)$, тогда алгоритм действует следующим образом. Если T_{h+1} не сохранен в T_{cache} , то он вычисляется и сохраняется. Затем извлекается уже описанным способом простой путь P из φ и добавляется в результирующее множество S . Далее алгоритм обрабатывает все ответвления.

1) Для дуги $e = (u, v) \in E, u \in P_{vht}^{T_{h+1}}, v \notin \{s, \dots, v_h, \dots, u\}$ проверяем, является ли простым путь P_e , следующий сначала вдоль P от вершины s до v_h , затем вдоль $P_{v_h u}^{T_{h+1}}$, вдоль ребра e и, наконец, вдоль $P_{vt}^{T_{h+1}}$.

2) Если P_e является простым, то добавляем в C_{simple} элемент вида $((T_0, e_0, T_1, \dots, T_h, e_h, T_{h+1}, e, T_{h+1}), lb_e)$, где lb_e – длина пути P_e .

3) Если ответвление e не является простым, то оно добавляется в множество $X = \{f_1, \dots, f_r\}$, такое что для всех $i, j, l : 1 \leq i < j < l \leq r$ $tail(f_j)$ находится между (или равен) $tail(f_i)$ и $tail(f_l)$ на пути $P_{vht}^{T_{h+1}}$.

4) Если множество $X \neq \emptyset$, то необходимо добавить X как группу кандидатов в $C_{not-simple}$.

Пусть $f_i = (u_i, v_i)$, тогда обозначим $lb_i = lb_{f_i} = w_G(P_1) + w_G(P_{v_h u_i}^{T_{h+1}}) + w_G(f_i) + w_G(P_{v_i t}^{T_{h+1}})$, в этом случае $lb = \min_{1 \leq i \leq r} lb_i$, где P_1 путь от s до v_h вдоль P .

В случае, если элемент φ извлекается из очереди $C_{not-simple}$, то φ имеет вид $((T_0, e_0, T_1, \dots, T_h, e_h, T_{h+1}, X, T_{h+1}), lb)$, $e_h = (u_h, v_h')$ и $P_1 = (s = x_0, \dots, x_h = v_h')$ – префикс пути, представленного элементом φ , $P_{v_h' t}^{T_{h+1}} = (v_h', v_{h+1}', \dots, v_p' = t)$, и для $\forall j$ $f_j = (v_{i_j}', v_j')$. $\exists i^* : 1 \leq i^* \leq r$, такой что $lb = lb_{i^*}$, алгоритм обрабатывает одновременно несколько ответвлений $f_r, f_{r-1}, \dots, f_{i^*}$.

1) Строится обратное дерево кратчайших путей T_r' в графе $G_r = G - \{x_0, \dots, x_h = v_h', \dots, v_{i_r}'\}$ с использованием алгоритма Дейкстры. В очередь C_{simple} добавляется элемент

вида $((T_0, e_0, T_1, \dots, T_h, e_h, T_{h+1}, f_r, T'_r), w_G(Q_r))$, где $Q_r = P_1 P_{v'_h v'_{i_r}}^{T_{h+1}} P_{v_r t}^{T'_r}$ — простой путь от s до t . Дерево T'_r не сохраняется в T_{cache} , но сохраняется «указатель» на него.

2) Для каждого j от $r - 1$ до i^* (по убыванию) строится дерево T'_j в графе $G_r = G - \{x_0, \dots, x_h = v'_h, \dots, v'_{i_r}\}$ на основе T'_{j+1} с использованием алгоритма обновления обратного дерева кратчайших путей [12]. В очередь C_{simple} добавляется элемент вида $((T_0, e_0, T_1, \dots, T_h, e_h, T_{h+1}, f_j, T'_j), w_G(Q_j))$, где $Q_j = P_1 P_{v'_h v'_{i_j}}^{T_{h+1}} P_{v_j t}^{T'_j}$.

Оставшаяся же часть ответвлений $((T_0, e_0, T_1, \dots, T_h, e_h, T_{h+1}, X', T_{h+1}), \min_{1 \leq j < i^*} lb_j)$, где $X' = \{f_1, \dots, f_{i^*-1}\}$ возвращается в $C_{not-simple}$.

Сложность алгоритма в худшем случае аналогична сложности базовой версии — алгоритма Йена, то есть $O(kn(m + n \log n))$.

2. Численные эксперименты.

2.1. *Описание графов.* Для тестирования алгоритмов использованы графы, полученные из различных источников [13–15].

— **NW** (USA-road-d/USA-road-t). Дорожная сеть Северо-Запада США. Граф ориентированный, дуги соответствуют направлениям движения, веса — расстояниям. Размеры: 1 207 945 вершин и 2 840 208 дуг [13].

— **FLA** (USA-road-d/USA-road-t). Дорожная сеть штата Флорида. Граф аналогичен указанному выше. Размеры: 1 070 376 вершин и 2 712 798 дуг [13].

— **Rome99**. Большой фрагмент городской дорожной сети Рима (Италия). Размеры: 3 353 вершин и 8 870 дуг [13].

— **14** (SNAP). Сеть соавторств в категории *Astro Physics* на arXiv: неориентированный граф, ребро между авторами означает совместную публикацию. Размеры: 18 772 вершин и 198 110 ребер [14].

— **15** (SNAP). Сеть соавторств в категории *Condensed Matter* (arXiv), неориентированный граф. Размеры: 23 133 вершин и 93 497 ребер [15].

— **16** (SNAP). Сеть соавторств в категории *General Relativity and Quantum Cosmology* (arXiv), неориентированный граф. Размеры: 5 242 вершин и 14 496 ребер [16].

— **17** (SNAP). Сеть соавторств в категории *High Energy Physics – Theory* (arXiv), неориентированный граф. Размеры: 9 877 вершин и 25 998 ребер [17].

2.2. *Результаты вычислений.* Результаты работы алгоритма были протестированы на графах, описанных выше.

В табл. 1, 2 приведены время выполнения и объем памяти, использованной при выполнении алгоритма поиска k кратчайших путей для 1000 произвольных пар вершин при различных значениях k .

В табл. 1 сделан акцент на ускорение по времени, при этом увеличивается объем использованной памяти. В табл. 2 реализован оптимальный алгоритм, в котором была поставлена цель уменьшить затраты памяти, при этом увеличилось время исполнения. Для графов sa-AstroPh и sa-CondMat в табл. 1 получена и оптимизация по памяти.

Реализованный оптимизированный алгоритм поиска k кратчайших путей показал хороший результат. В случае, когда графы небольшие, время с увеличением k растет незначительно. Для первых двух графов, в которых количество вершин больше миллиона, с увеличением k растет и время.

Заключение. Алгоритм использовался для реализации поиска глобальных путей по графам элементов в задачах химической кинетики. Глобальным путем в заданном графе потока элемента является путь от вещества в исходной смеси до конечного продукта,

Таблица 1

Оптимизация по времени

Граф	k=2		k=3		k=4		k=5		k=10	
	Т, мс	М, Кб	Т, мс	М, Кб	Т, мс	М, Кб	Т, мс	М, Кб	Т, мс	М, Кб
NW	148	45227	208	48119	234	49409	314	53414	556	65846
FLA	127	41065	172	43385	227	45095	304	48684	595	60721
Rome99	0,01	1813	0,03	1836	0,1	1857	0,19	1888	0,77	2019
ca-AstroPh	3,6	2683	3,3	2706	3,6	2726	3,5	2750	3,9	2859
ca-CondMat	2,6	2792	2,6	2792	4	2746	4,6	2759	3,6	2799
ca-GrQc	0,002	1821	0,002	1827	0,001	1831	0,003	1810	0,015	1817
ca-НepTh	0,71	2091	0,67	2098	0,69	2113	0,67	2114	0,7	2127

Таблица 2

Оптимизация по памяти

Граф	k=2		k=3		k=4		k=5		k=10	
	Т, мс	М, Кб	Т, мс	М, Кб	Т, мс	М, Кб	Т, мс	М, Кб	Т, мс	М, Кб
NW	253	32690	365	34536	450	35336	591	37058	1225	46079
FLA	281	30406	374	32204	454	33312	693	35842	1148	44361
Rome99	0,08	1596	0,2	1598	0,39	1607	0,64	1615	1,47	1705
ca-AstroPh	8,7	2905	8,4	2897	8,4	2899	8,6	2908	9,2	2932
ca-CondMat	8,7	3004	8,8	3024	8,7	3015	8,9	3020	8,6	3024
ca-GrQc	0,7	1750	0,7	1751	0,7	1748	0,7	1746	0,7	1741
ca-НepTh	2,3	1991	2,4	1994	2,5	2001	2,4	2005	2,4	2009

проходящий через вещество-концентратор. Глобальный путь позволяет идентифицировать последовательные цепочки превращения соединений от начальных реагентов до конечных продуктов всего реакционного процесса. Учитывая основные глобальные пути, через которые проходят наибольшие потоки элемента, можно идентифицировать наиболее важные цепочки превращения, которые должны быть сохранены в скелетном механизме.

Для каждого вещества-концентратора существует множество глобальных путей, поэтому для идентификации основных глобальных путей используется алгоритм нахождения k кратчайших независимых путей в ориентированном взвешенном графе. Глобальный путь вычисляется как комбинация пути от вещества из исходной смеси до вещества-концентратора с путем от вещества-концентратора до одного из продуктов. На выходе получается множество мощностью k^2 глобальных путей. Все такие пути для каждого вещества-концентратора и каждой пары заданных соединений в каждом графе потока элемента сохраняются и образуют множество веществ, входящих в итоговый скелетный механизм.

В работе рассмотрена реализация алгоритма для дорожных сетей с большим количеством путей, и проведена оптимизация, которая позволила сократить ресурсы по времени и памяти.

В дальнейшем алгоритм будет использоваться в качестве одного из методов исследования и построения фактор-деревьев и древовидных разбиений графа для ускорения времени решения систем линейных уравнений.

Список литературы

1. Egli W., Kraus M. Shortest paths in chemical kinetic applications // Phys. Chem. Chem. Phys. 2003. Vol. 5. P. 3916–3920.
2. Yen Jin Y., Finding the K shortest loopless paths in a network // Manag. Sci. 1971. Vol. 17. P. 712–716.
3. Hoffman R., Pavley R. A method for the solution of the nth best path problem // J. of the Association for Computing Machinery. 1959. Vol. 6, № 4. P. 506–515.
4. Bellman R., Kalaba R. On kth best policies // J. of SIAM. 1960. Vol. 8, № 4. P. 582–588.
6. Bock F., Kanther H., Haynes J. An algorithm (the rh best path algorithm) for finding and ranking paths through a network // Research Report, Armour Research Foundation. Chicago, Illinois, 1957.
7. Clarke S., Krikorian A., Rausan J. Computing the N best loopless paths in a network // J. of SIAM. 1963. Vol. 11, № 4. P. 1096–1102.
9. Dijkstra E. W. A note on two problems in connexion with graphs // Numerische Mathematik. 1959. Vol. 1. P. 269–271.
8. Pollack M. The kth best route through a network // Operations Research. 1961. Vol. 9, № 4. P. 578.
5. Sakarovitch M. The k shortest routes and the k shortest chains in a graph // Operations Research Center Report № 66–32. University of California, Berkeley. 1966.
10. Coudert D., Al Zoobi A., Nisse N. Space and time trade-off for the k shortest simple paths problem space and time trade-off for the k shortest simple paths problem // 18th International Symposium on Experimental Algorithms (SEA 2020). Art. № 18. P. 18:1–18:13.
11. Kurz D., Mutzel P. A sidetrack-based algorithm for finding the k shortest simple paths in a directed graph // In Int. Symp. on Algorithms and Computation (ISAAC), Schloss Dagstuhl, 2016. Vol. 64 of LIPIcs. P. 49:1–49:13.
12. Frigioni D., Marchetti-Spaccamela A., Nanni U. Fully dynamic algorithms for maintaining shortest paths trees // J. of Algorithms. 2000. Vol. 34, № 2. P. 251–281.
13. [Электрон. рес.]: <https://www.diag.uniroma1.it/challenge9/download.html>.
14. [Электрон. рес.]: <https://snap.stanford.edu/data/ca-AstroPh.html>.
15. [Электрон. рес.]: <https://snap.stanford.edu/data/ca-CondMat.html>.
16. [Электрон. рес.]: <https://snap.stanford.edu/data/ca-GrQc.html>.
17. [Электрон. рес.]: <https://snap.stanford.edu/data/ca-HepTh.html>.



Герб Артем Родионович — аспирант Института вычислительной математики и математической геофизики СО РАН. В 2023 году окончил механико-математический факультет НГУ, степень магистра. E-mail: gerb-artem@mail.ru.

ru.

Gerb Artem is a postgraduate student of the Institute of Computational Mathematics and Mathematical Geophysics SB RAS. Graduated from Novosibirsk State University's Math Faculty in 2023.

Девятых Егор Евгеньевич — студент 1 курса магистратуры механико-математического факультета Новосибирско-

го государственного университета. E-mail: klasifate@gmail.com.



Devyatykh Egor is a 1st year master's student of the Faculty of Mechanics and Mathematics of Novosibirsk State University.

Омарова Гульзира Алимовна — канд. физ.-мат. наук, науч. сотр. Института вычислительной математики и математической геофизики СО РАН; e-mail: gulzira@rav.sssc.ru.



Omarova Gulzira is PhD in Physics and Mathematics, Research Fellow at the Institute of Computational Mathematics and Mathematical Geophysics SB RAS.

Дата поступления — 15.12.2025