УДК 519.168

В.А. ВАСЯНИН

ДВУХКРИТЕРИАЛЬНЫЙ ЛЕКСИКОГРАФИЧЕСКИЙ АЛГОРИТМ ПОСТРОЕНИЯ ВСЕХ КРАТЧАЙШИХ ПУТЕЙ В СЕТИ

Аннотация. Рассматривается алгоритм построения кратчайших путей между всеми парами узлов в неориентированной сети по критерию: минимум дуг в пути; минимум длины пути. Проведен анализ трудоемкости алгоритма и эмпирически показано, что по мере увеличения плотности сети его вычислительная эффективность становится выше, чем у алгоритма Флойда, соответствующим образом модифицированного для нахождения кратчайших путей по ступенчатому критерию.

Ключевые слова: многокритериальные задачи построения кратчайших путей, алгоритмы, вычислительная трудоемкость.

ВВЕДЕНИЕ

Во многих теоретических и прикладных задачах оптимизации на графах, таких, например, как проектирование транспортных, информационных и телекоммуникационных сетей, требуется построение кратчайших путей (КП). В большей части публикаций, посвященных вопросам нахождения КП, рассматриваются задачи построения КП по одному критерию — минимум или максимум суммы длин дуг в пути. Под длиной дуги понимается некоторый ее параметр, например расстояние по дуге, затраты при перевозке по дуге, пропускная способность дуги и т.п.

Наряду с задачами построения КП по одному критерию значительный интерес представляют многокритериальные задачи о путях (МКП), естественным образом возникающие во многих приложениях, когда имеется несколько различных параметров, характеризующих узлы и дуги сети. В большинстве случаев выбор оптимальных кратчайших путей по экстремальным значениям этих параметров представляет сложную задачу, поскольку параметры могут быть противоречивыми и конкурировать между собой. Концепция оптимизации МКП задачи в целом, в отличие от задачи оптимизации путей по одному критерию, заключается в необходимости нахождения некоторого оптимального решения по всем заданным критериям. В этом случае, как правило, существует не одно, а множество парето-оптимальных решений задачи, удовлетворяющих всем критериям.

Поскольку обзор современных многокритериальных алгоритмов представляет отдельную тему, выходящую за рамки настоящей статьи, отметим лишь, что они исследовались менее досконально, например работа Хансена [1], опубликованная в 1979 г. и посвященная систематическому исследованию двухкритериальных задач о путях. Хансен [1] и независимо Варбартон [2] разработали полностью полиномиальную схему решения МКП задачи (fully polynomial time approximation schemes — FPTAS) для парето-оптимальных решений в случае двух критериев. Двухкритериальные задачи также были достаточно исследованы в [3]. После этих публикаций появилось достаточно много работ, связанных с многокритериальными задачами о КП, в которых исследовались вопросы разработки эффективных алгоритмов нахождения парето-оптимальных путей по нескольким конкурирующим между собой критериям. Для случаев, когда число критериев оптимизации больше двух, пока не достигнуто особых результатов. Судя по литературным источникам и публикациям в Интернете, особых продвижений в решении МКП задач с использованием FPTAS-схем пока не наблюдается.

В настоящей статье рассматриваются алгоритмы построения КП, упорядоченных в лексикографическом порядке по двум критериям. Известно, что такие алгоритмы имеют полиномиальную трудоемкость в отличие от алгоритмов нахождения парето-оптимальных кратчайших путей. Некоторые алгоритмы построения ступенчатых двухкритериальных путей рассматривались в работах [4, 5]. Идея алгоритма, предложенного в [4], близка к идее алгоритма Мура [6] и отличается особенностью реализации процедуры расстановки «пометок» с помощью логических операций над строками матриц, характеризующих сеть. Как показано в [7], асимптотическая трудоемкость алгоритмов такого типа составляет $O(n^3)$, где n — число узлов в сети. В работе [5] изложен алгоритм, строящий деревья кратчайших путей по двухступенчатому критерию: минимум ранга пути, при равенстве рангов — максимум пропускной способности (под рангом пути понимается число транзитных узлов или дуг в пути). Суть алгоритма заключается в следующем: от некоторого узла i необходимо построить дерево КП до всех остальных узлов. Поскольку пути с рангом 1 уже построены, на первой итерации алгоритма строятся пути с рангом 2. На каждой последующей итерации строятся пути с рангом, большим на единицу предшествующего ранга. Итерация включает два вложенных цикла по числу узлов в сети, поэтому асимптотическая трудоемкость итерации составляет $O(n^2)$. Для построения дерева КП от узла i потребуется $O((q_i - 1)n^2)$ действий, где q_i — максимальный ранг КП, построенного от узла i. Нахождение деревьев КП от всех узлов потребует сложности порядка $O((q_{\max} - 1)n^3)$ в предположении, что для всех узлов имеем $q_i = q_{\max}$.

Следует особо отметить, что для построения путей по ступенчатому критерию легко модифицировать любой известный алгоритм построения КП по одному критерию, например алгоритм Флойда [8] или Дейкстры [9]. Как известно, эти алгоритмы имеют асимптотическую трудоемкость порядка $O(n^3)$ для построения КП от всех узлов.

Как правило, при решении большинства практических задач нахождения КП по двум критериям наибольший интерес представляют алгоритмы, использующие в качестве критериев оптимизации минимум транзитных узлов или дуг в пути (ранг пути) и минимум длины пути. В связи с этим в настоящей работе рекомедуется улучшенная версия двухкритериального алгоритма построения всех КП в сети по указанным критериям, предложенного в работе [10]. Поскольку в настоящей статье рассматривается построение КП от всех узлов в сети, трудоемкость предложенного алгоритма будет сравниваться с трудоемкостью измененного по таким же критериям алгоритма Флойда. Следует отметить, что приведенные алгоритмы могут быть легко обобщены и на случай наличия большего числа упорядоченных критериев оптимизации.

ПОСТАНОВКА ЗАДАЧИ И АЛГОРИТМ РЕШЕНИЯ

Пусть задана простая неориентированная сеть $G(N, P)$ с множеством узлов N, $n = |N|$, и множеством дуг P, $p = |P|$. Каждой дуге $p_{kl} \in P$ поставлено в соответствие некоторое число $r_{kl} > 0$, условно называемое ее длиной. Назовем число дуг, составляющих некоторый путь, рангом пути. Требуется построить КП между всеми узлами сети, удовлетворяющие критерию: минимум ранга пути, при равенстве рангов — минимум длины.

Идея предлагаемого алгоритма сходна с идеей алгоритма Беллмана–Шимбела [11, 12], согласно которой на каждой условной l-й итерации строятся все КП в сети с рангом до 2^l включительно. При этом на каждой последующей итерации используются все пути, построенные на предыдущих, и применяется эффективная техника представления абстрактных типов данных (АТД).

Пусть $R = \| r_{ij} \|_{n \times n}$ — топологическая матрица; r_{ij} — длина дуги p_{ij} (если дуги p_{ij} не существует, $r_{ij} = \infty$); $D = \| d_{ij} \|_{n \times n}$ — матрица длин построенных путей; $C = \| c_{ij} \|_{n \times n}$ — справочная матрица построенных путей, каждый элемент которой c_{ij}, $i \neq j$, определяет номер предпоследнего узла на кратчайшем пути от i до j, $c_{ii} = 0$, $i = \overline{1, n}$; $Q = \| q_{ij} \|_{n \times n}$ — матрица рангов построенных путей. Первоначально D совпадает с R, а для путей с рангом 1 соответствующие элементы матриц $q_{ij} = 1$, $c_{ij} = i$. Для ненайденных путей имеем $q_{ij} = \infty$, $c_{ij} = 0$. Знаки $\leftarrow$, $\wedge$ и $\vee$ соответственно означают операции присваивания, конъюнкции (логического «и») и дизъюнкции (логического «или»). Знаком *** отделяются комментарии.

Приведем алгоритм построения КП по ступенчатому критерию [10].

Алгоритм SP1

1. $RMAX \leftarrow 1$; $l \leftarrow 0$.
2. $KU \leftarrow 0$; $l \leftarrow l + 1$.
3. Для $\{i \mid i = \overline{1, n}\}$ выполнить пп. 4–16.
4. $KS \leftarrow 0$.
5. Для $\{j \mid j = \overline{1, n}\}$ выполнить пп. 6–14.
6. Если $(i \neq j) \wedge c_{ij} = 0$, то перейти к п. 7; иначе выполнить $KS \leftarrow KS + 1$ и перейти к п. 14.
7. Для $\{k \mid k = \overline{1, n}\}$ выполнить пп. 8–12.
8. Если $Q(i, k) \leq RMAX \wedge Q(k, j) \leq RMAX$, то перейти к п. 9; иначе перейти к п. 12.
9. $RT \leftarrow Q(i, k) + Q(k, j)$.
10. Если $RT < Q(i, j) \vee (RT = Q(i, j) \wedge D(i, j) > D(i, k) + D(k, j))$, то перейти к п. 11; иначе перейти к п. 12.
11. $D(i, j) \leftarrow D(i, k) + D(k, j)$; $Q(i, j) \leftarrow RT$; $C(i, j) \leftarrow C(k, j)$.
12. Перейти к п. 7. *** Конец цикла по k
13. Если $C(i, j) \neq 0$, то $KS \leftarrow KS + 1$.
14. Перейти к п. 5. *** Конец цикла по j
15. Если $KS = n$, то $KU \leftarrow KU + 1$.
16. Перейти к п. 3. *** Конец цикла по i
17. $RMAX \leftarrow 2RMAX$.
18. Если $RMAX \leq (n-1) \wedge KU < n$, то перейти к п. 2, иначе к п. 19.
19. Если $RMAX > (n-1) \wedge KU < n$, то вывести сообщение о несвязности сети.
20. Конец.

Приведем измененный алгоритм Флойда для построения КП по ступенчатому критерию.

Алгоритм Floyd

1. Для $\{i \mid i = \overline{1, n}\}$ выполнить пп. 2–9.
2. Для $\{j \mid j = \overline{1, n}\}$, если $i \neq j$, выполнить пп. 3–8.
3. Для $\{k \mid k = \overline{1, n}\}$, если $k \neq j \neq i$, выполнить пп. 4–7.
4. Если $Q(j, i) < \infty \wedge Q(i, k) < \infty$, то перейти к п. 5; иначе перейти к п. 7.
5. Если $(Q(j, k) > Q(j, i) + Q(i, k)) \vee (Q(j, k) = Q(j, i) + Q(i, k) \wedge D(j, k) > D(j, i) + D(i, k))$, то перейти к п. 6; иначе перейти к п. 7.
6. $Q(j, k) \leftarrow Q(j, i) + Q(i, k)$; $D(j, k) \leftarrow D(j, i) + D(i, k)$; $C(j, k) \leftarrow C(i, k)$.
7. Перейти к п. 3.
8. Перейти к п. 2.

9. Перейти к п. 1.

10. Конец.

В алгоритме SP1 в строке 2 начинается цикл, представляющий условные итерации, общее число которых для построения всех путей в сети накапливается в l. Условная итерация содержит три вложенных цикла по индексам: i, j, k.

Предположим, что каждому номеру l, $l = 0, 1, 2, \ldots$, условной итерации алгоритма SP1 поставлено во взаимно-однозначное соответствие множество $\{1+2^l, \ldots, 2^{l+1}\}$ рангов путей. Назовем это соответствие таблицей рангов. Корректность алгоритма следует из доказанных в [10] следующих утверждений.

Лемма. Пусть в алгоритме SP1 на l-й итерации допускается построение путей с рангом, большим 2^l, тогда кратчайшие пути будут построены некорректно.

Доказательство. В строке 8 алгоритма SP1 во внутреннем цикле по k возможно $Q(i, k) < \infty$ и $Q(k, j) < \infty$. Пусть на некоторой l-й итерации построены пути (i, k_1), (i, k_2), причем $2^l < Q(i, k_2) = Q(i, k_1) + Q(k_1, k_2) < \infty$ и номер k_2 больше номера k_1. Поскольку в цикле по k осуществляется перебор всех номеров узлов в возрастающем порядке, то найдется узел с номером $k_3 > k_2 > k_1$ такой, что

$$\begin{gathered} Q(i, k_2) = Q(i, k_3) + Q(k_3, k_2), \\ D(i, k_2) = D(i, k_3) + D(k_3, k_2) < D(i, k_2) = D(i, k_1) + D(k_1, k_2). \end{gathered} \tag{1}$$

Однако к моменту определения пути до k_2 имеем $Q(i, k_3) = \infty$ или $Q(k_3, k_2) = \infty$, т.е. путь (i, k_3) или (k_3, k_2) еще не найден. На l-й итерации путь (i, k_2) получил отметку $C(i, k_2) = k_1$ и на последующих итерациях рассматриваться не будет; в силу выполнения (1) этот путь некорректен.

Теорема 1. На каждой условной итерации алгоритма SP1 строятся все кратчайшие пути в сети, соответствующие таблице рангов.

Доказательство. Поскольку на каждой l-й итерации алгоритма нельзя строить пути с рангом, большим 2^l (лемма), для построения путей на этой итерации используем только те пути, которые построены на $(l-1)$-й итерации. Фактически это гарантируется использованием в цикле по k в строке 8 следующих операций сравнения: $Q(i, k) \le RMAX$, $Q(k, j) \le RMAX$, $RMAX = 2^l$, $l = 0, 1, \ldots$, а первой условной итерации соответствует номер $l = 0$. Очевидно, что на каждой l-й итерации алгоритма будут построены все пути в сети с рангами, строго соответствующими таблице рангов. Докажем, что построенные пути являются кратчайшими. Пусть на m-й итерации получен путь (i, j), не удовлетворяющий критерию: минимум ранга, минимум длины. Значит, существует другой кратчайший путь, который проходит через узел u, причем либо $Q(i, u)$, либо $Q(u, j)$ не определены. Поскольку на m-й итерации используются все пути, построенные на $(m-1)$-й итерации, во внутреннем цикле по k выполняется исчерпывающий поиск путей между i и j среди всех путей с множеством рангов $\{1+2^m, \ldots, 2^{m+1}\}$ по ступенчатому критерию:

$$t = \arg\min \{Q(i, k) + Q(k, j)\}, \;\; k = \overline{1, n}, \;\; k \ne u;$$

$$D(i, j) = \min \{D(i, v) + D(v, j)\}, \;\; v \in t,$$

где t — множество узлов, через которые пути имеют одинаковый ранг. Следовательно, путь (i, j) через узел u мог быть построен только на последующих итерациях и имел бы больший ранг, что противоречит предположению о существовании другого кратчайшего пути. Теорема доказана.

Следствие. Для построения оптимального пути с рангом m $(m \ge 2)$ требуется $\lceil \log_2 m \rceil$ условных итераций алгоритма SP1, где символ $\lceil \cdot \rceil$ означает округление числа до большего целого.

Пусть $S = \lceil \log_2 m \rceil$, где m — максимальный ранг пути среди всех кратчайших путей, построенных в сети.

Теорема 2. Время, необходимое алгоритму SP1 для построения всех кратчайших путей в сети, возрастает пропорционально функции $Sn^3 - n^2(3S+c) + 2n(S+c)$, где c — некоторая константа.

Оценим трудоемкость алгоритма SP1. Обозначим $h_{k,i}$ количество путей, построенных от узла i и имеющих ранг k. Тогда для построения всех кратчайших путей в сети число выходов алгоритма на внутренний цикл по k определится следующим образом:

$$\sum_{i=1}^{n} n-(1+h_{1,i})+\sum_{i=1}^{n}(n-(1+h_{1,i}+h_{2,i}))+\ldots+\sum_{i=1}^{n}(n-(1+h_{1,i}+h_{2,i}+h_{3,i}+\ldots+h_{2^{(S-1)},i}))=$$

$$=\sum_{i=1}^{n}(n-(1+h_{1,i})+n-(1+h_{1,i}+h_{2,i})+\ldots+n-(1+h_{1,i}+h_{2,i}+h_{3,i}+\ldots+h_{2^{(S-1)},i}))=$$

$$=\sum_{i=1}^{n}(nS-(S+h_{1,i}S+h_{2,i}(S-1)+h_{3,i}(S-2)+h_{4,i}(S-2)+\ldots+h_{2^{(S-1)},i}))=$$

$$=\sum_{i=1}^{n}\left[(nS-S)-\sum_{l=0}^{S-1}(S-l)\sum_{k\in\{\lfloor 1+2^{(l-1)}\rfloor,\ldots,2^l\}} h_{k,i}\right]=$$

$$=Sn(n-1)-\sum_{i=1}^{n}\sum_{l=0}^{S-1}(S-l)\sum_{k\in\{\lfloor 1+2^{(l-1)}\rfloor,\ldots,2^l\}} h_{k,i},$$

где символ $\lfloor\cdot\rfloor$ означает округление числа до меньшего целого.

Число выполнений цикла по k зависит от структуры исходной сети, и с увеличением числа условных итераций алгоритма по l стремится к $(n-2)$. Поэтому дадим оценку сверху трудоемкости алгоритма в худшем случае:

$$T_{\text{тр}} = Sn(n-1)(n-2)-(n-2)\sum_{i=1}^{n}\sum_{l=0}^{S-1}(S-l)\sum_{k\in\{\lfloor 1+2^{(l-1)}\rfloor,\ldots,2^l\}} h_{k,i}. \qquad (2)$$

Обозначим $\sum_{l=0}^{S-1}(S-l)\sum_{k\in\{\lfloor 1+2^{(l-1)}\rfloor,\ldots,2^l\}} h_{k,i} = c$ в предположении, что $h_{k,i} = h_k$ для всех $i \in N$, тогда (2) перепишем в виде

$$T_{\text{тр}} = Sn(n-1)(n-2)-(n-2)\sum_{i=1}^{n} c = Sn^3 - n^2(3S+c)+2n(S+c). \qquad (3)$$

Теорема доказана.

Поскольку асимптотическая трудоемкость алгоритма Флойда составляет всегда $O(n^3)$, можно предполагать, что при увеличении плотности графа или сети быстродействие алгоритма SP1 будет увеличиваться в отличие от быстродействия алгоритма Флойда.

Из анализа алгоритма SP1 видно, что его трудоемкость можно уменьшить за счет сокращения перебора узлов j в цикле в строке 5 при выборе узлов, пути к которым еще не найдены, и узлов k в цикле в строке 7 при выборе узлов, пути к которым уже найдены. Введем следующие структуры данных. Пусть $A = \| a_{ij} \|_{n\times(n-1)}$ — матрица, содержащая номера узлов, пути к которым найдены. При этом в первой строке матрицы находятся номера узлов, к которым найдены пути от первого узла, во второй строке — от второго узла и т.д. Введем также вектор $T = || t_i ||_n$, где накапливается число узлов, к которым найдены пути от узлов i, $i = \overline{1, n}$, а также вектор $SP = || sp_i ||_n$, содержащий указатели sp_i от узлов i, $i = \overline{1, n}$, на линейные односторонние списки, состоящие из элементов E. Каждый элемент E состоит из двух полей: F и AJ. Поле AJ содержит номер узла, к которому путь не найден, а поле F содержит ссылку на следующий элемент. Последний элемент в списке, как и пустой список, имеет нулевые ссылки (*null*-ссылки).

Приведем улучшенную версию алгоритма SP1 построения КП по ступенчатому критерию.

Алгоритм SP2

1. $T \leftarrow 0$.
2. Для $\{i \mid i = \overline{1, n}\}$ выполнить пп. 3–12.
3. Для $\{j \mid j = \overline{1, n}\}$, если $j \neq i$, выполнить пп. 4–11.
4. Если $D(i, j) < \infty$, то перейти к п. 5, иначе перейти к п. 6.
5. $T(i) \leftarrow T(i) + 1$; $A(i, T(i)) \leftarrow j$; перейти к п. 11.
6. Образовать новый элемент $E(P)$.
7. Если $SP(i).F \neq null$, то выполнить п. 8, иначе выполнить п. 9.
8. $P2 \Rightarrow P1$; $P1 \Rightarrow P$; $P2.F \Rightarrow P$; перейти к п. 10.
9. $SP(i).F \Rightarrow P$; $P1 \Rightarrow P$; перейти к п. 10.
10. $P.AJ \leftarrow j$.
11. Перейти к п. 3. *** Конец цикла по j
12. Перейти к п. 2. *** Конец цикла по i
13. $RMAX \leftarrow 1$; $l \leftarrow 0$.
14. $KU \leftarrow 0$; $l \leftarrow l + 1$.
15. Для $\{i \mid i = \overline{1, n}\}$ выполнить пп. 16–31.
16. Если $SP(i).F \neq null$, то перейти п. 17, иначе $KU \leftarrow KU + 1$; перейти к п. 31.
17. $P \Rightarrow SP(i).F$; $P1 \Rightarrow SP(i)$.
18. Пока $P.F \neq null$, выполнить пп. 19–30.
19. $j \leftarrow P.AJ$.
20. Для $\{m \mid m = \overline{1, T(i)}\}$ выполнить пп. 21–26.
21. $k \leftarrow A(i, m)$.
22. Если $Q(i, k) \leq RMAX \wedge Q(k, j) \leq RMAX$, то перейти к п. 23; иначе перейти к п. 26.
23. $RT \leftarrow Q(i, k) + Q(k, j)$.
24. Если $RT < Q(i, j) \vee (RT = Q(i, j) \wedge D(i, j) > D(i, k) + D(k, j))$, то перейти к п. 25; иначе перейти к п. 26.
25. $D(i, j) \leftarrow D(i, k) + D(k, j)$; $Q(i, j) \leftarrow RT$; $C(i, j) \leftarrow C(k, j)$.
26. Перейти к п. 20. *** Конец цикла по m
27. Если $C(i, j) \neq 0$, то перейти к п. 28, иначе перейти к п. 29.
28. $T(i) \leftarrow T(i) + 1$; $A(i, T(i)) \leftarrow j$; $P1.F \Rightarrow P.F$; $P3 \Rightarrow P$; $P \Rightarrow P.F$; Освободить элемент $E(P3)$. Перейти к п. 30.
29. $P1 \Rightarrow P$; $P \Rightarrow P.F$. Перейти к п. 30.
30. Перейти к п. 18. *** Конец цикла по j
31. Перейти к п. 15. *** Конец цикла по i
32. $RMAX \leftarrow 2RMAX$.
33. Если $RMAX \leq (n-1) \wedge KU < n$, то перейти к п. 14, иначе к п. 34.
34. Если $RMAX > (n-1) \wedge KU < n$, то вывести сообщение о несвязности сети.
35. Конец.

В строках 1–12 выполняется формирование начальных значений для T, A, SP, а в строках 13–35 содержится основное тело алгоритма. Вектор SP используется во внешних циклах по i и j для выбора узлов j, к которым пути еще не найдены (строки 15–31 и 18–30). Матрица A и вектор T используются во внутреннем цикле по m для выбора номеров узлов k, к которым пути от i до k уже найдены (строки 20–26). Если на данной итерации найден КП от i до j через какой-либо узел k, то j вводится в соответствующую i-ю строку A, а из списка sp_i j-й узел выводится. По окончании работы алгоритма списки sp_i становятся пустыми, а строки матрицы A будут полностью заполнены (в случае, если сеть связна).

Операция «Образовать новый элемент $E(P)$» в п. 6 означает образование элемента E и установку на него указателя P, а операция «Освободить элемент $E(P3)$» в п. 28 означает удаление элемента E, на который ссылается указатель $P3$. Операции со знаком $\Rightarrow$ устанавливают указатель в левой части выражения на элемент в правой части выражения. Выражения с точками типа $SP(i).F$, $P.F$, $P.AJ$ означают соответствующие поля элементов, на которые ссылаются указатели $SP(i)$ и P. Если в строке 33 окажется, что $RMAX > (n-1)$, то при $KU < n$ исходная сеть несвязна.

Для алгоритмов SP1, Floyd и SP2 были разработаны модифицированные алгоритмы SP1S, FloydS и SP2S для случая, когда матрица R является симметричной, так как свойство симметричности позволяет существенно сократить время нахождения КП.

Измененные строки для алгоритма SP1S:

3. Для $\{i \mid i = \overline{1, n-1}\}$ выполнить пп. 4–16.

5. Для $\{j \mid j = \overline{i+1, n}\}$ выполнить пп. 6–14.

6. Если $c_{ij} = 0$, то перейти к п. 7; иначе выполнить $KS \leftarrow KS + 1$ и перейти к п. 14.

11. $D(i, j) \leftarrow D(i, k) + D(k, j)$; $D(j, i) \leftarrow D(i, j)$; $Q(i, j) \leftarrow RT$; $Q(j, i) \leftarrow RT$; $C(i, j) \leftarrow C(k, j)$; $C(j, i) \leftarrow C(k, i)$.

15. Если $KS = n - i$, то $KU \leftarrow KU + 1$.

18. Если $RMAX \le (n-1) \wedge KU < (n-1)$, то перейти к п. 2, иначе к п. 19.

19. Если $RMAX > (n-1) \wedge KU < (n-1)$, то вывести сообщение о несвязности сети.

Измененные строки для алгоритма FloydS:

2. Для $\{j \mid j = \overline{1, n-1}\}$, если $i \ne j$, выполнить пп. 3–8.

3. Для $\{k \mid k = \overline{j+1, n}\}$, если $k \ne j \ne i$, выполнить пп. 4–7.

6. $Q(j, k) \leftarrow Q(j, i) + Q(i, k)$; $Q(k, j) \leftarrow Q(j, k)$; $D(j, k) \leftarrow D(j, i) + D(i, k)$; $D(k, j) \leftarrow D(j, k)$; $C(j, k) \leftarrow C(i, k)$; $C(k, j) \leftarrow C(i, j)$.

Измененные строки для алгоритма SP2S:

4. Если $D(i, j) < \infty$, то перейти к п. 5, иначе перейти к п. 5а.

5а. Если $i < j$, то перейти к п. 6, иначе к п. 11.

15. Для $\{i \mid i = \overline{1, n-1}\}$ выполнить пп. 16–31.

25. $D(i, j) \leftarrow D(i, k) + D(k, j)$; $D(j, i) \leftarrow D(i, j)$; $Q(i, j) \leftarrow RT$; $Q(j, i) \leftarrow RT$; $C(i, j) \leftarrow C(k, j)$; $C(j, i) \leftarrow C(k, i)$.

28. $T(i) \leftarrow T(i) + 1$; $T(j) \leftarrow T(j) + 1$; $A(i, T(i)) \leftarrow j$; $A(j, T(j)) \leftarrow i$; $P1.F \Rightarrow P.F$; $P3 \Rightarrow P$; $P \Rightarrow P.F$; Освободить элемент $E(P3)$. Перейти к п. 30.

33. Если $RMAX \le (n-1) \wedge KU < (n-1)$, то перейти к п. 14, иначе к п. 34.

34. Если $RMAX > (n-1) \wedge KU < (n-1)$, то вывести сообщение о несвязности сети.

Для проведения эксперимента была составлена тестовая программа на языке Digital Visual Fortran (DVF) компании Digital Equipment Corporation в среде Microsoft (MS) Developer Visual Studio (VS) DVF 6.1. Во внешней программе в режиме диалога вводились: n — число узлов сети; val — число исходящих из каждого узла дуг (степень узла); границы изменения значений весов дуг от $MINVAL$ до $MAXVAL$; параметр, управляющий выводом входных и выходных данных. Далее с помощью датчика случайных чисел (встроенной функции языка $RAND(\,)$) генерировались веса дуг от $MINVAL$ до $MAXVAL$, формировались массивы R, D, Q, C, а также массивы A, T, SP (для алгоритмов SP2, SP2S). Вся оперативная память, необходимая для работы алгоритмов, выделялась и освобождалась динамически во внешней программе. Время работы алгоритмов фиксировалось встроенной подпрограммой *cputime*() непосредственно до входа и после выхода из алгоритмов построения КП. Работа алгоритмов проверялась на ПЭВМ IP-IV с тактовой частотой 2,66 ГГц и оперативной памятью 2 Гб под управлением операционной системы Windows Vista. Отдельно проводилось моделирование решения задачи на разреженных сетях с числом узлов $n=1000$ при изменении значений параметра val от 2 до 999; при изменении n от 100 до 1000 при $val=5$; на сильно разреженных сетях при $val=2$ и изменении n от 500 до 1000. Во всех случаях принималось $MINVAL=30$, $MAXVAL=120$. Все алгоритмы были оформлены в виде внешних подпрограмм с передачей фактических параметров.

На рис. 1 приведены соответственно графики изменения времени работы алгоритмов SP1, Floyd, SP2 и SP1S, FloydS, SP2S в зависимости от роста степени узлов val от 2 до 999 для $n=1000$.

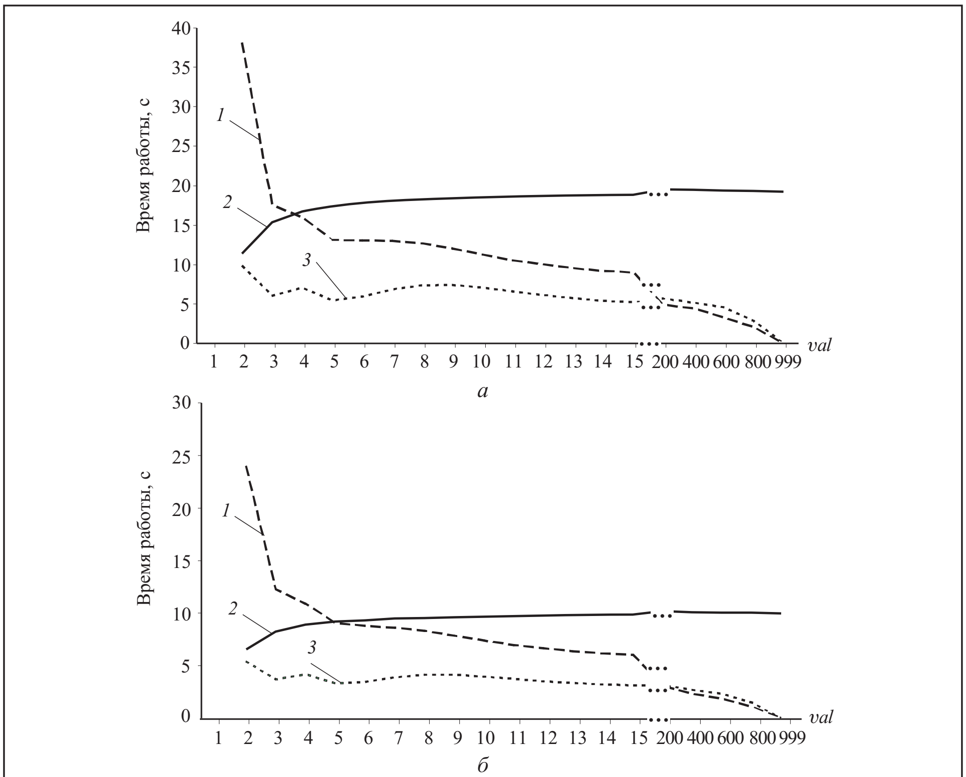

Рис. 1. График изменения времени работы алгоритмов: SP1 (*1*), Floyd (*2*), SP2 (*3*) (*а*); SP1S (*1*), FloydS (*2*), SP2S (*3*) (*б*) в зависимости от степени узлов

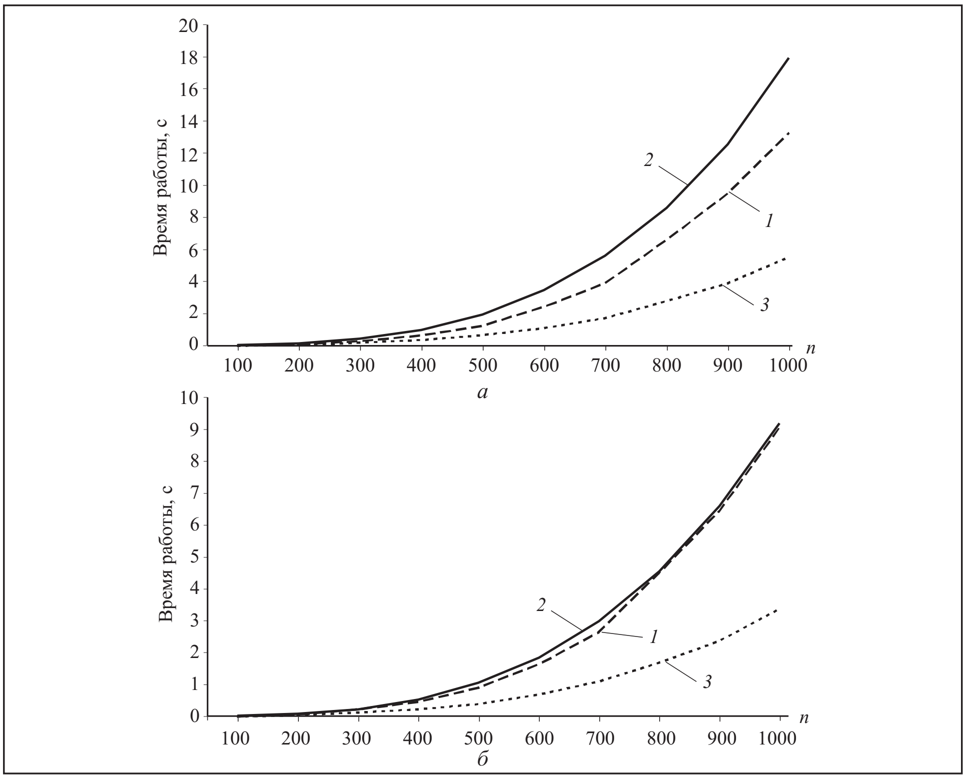

Рис. 2. График изменения времени работы алгоритмов SP1 (*1*), Floyd (*2*), SP2 (3) (*а*) и SP1S (*1*), FloydS (*2*), SP2S (*3*) (*б*) в зависимости от роста числа узлов при $val = 5$

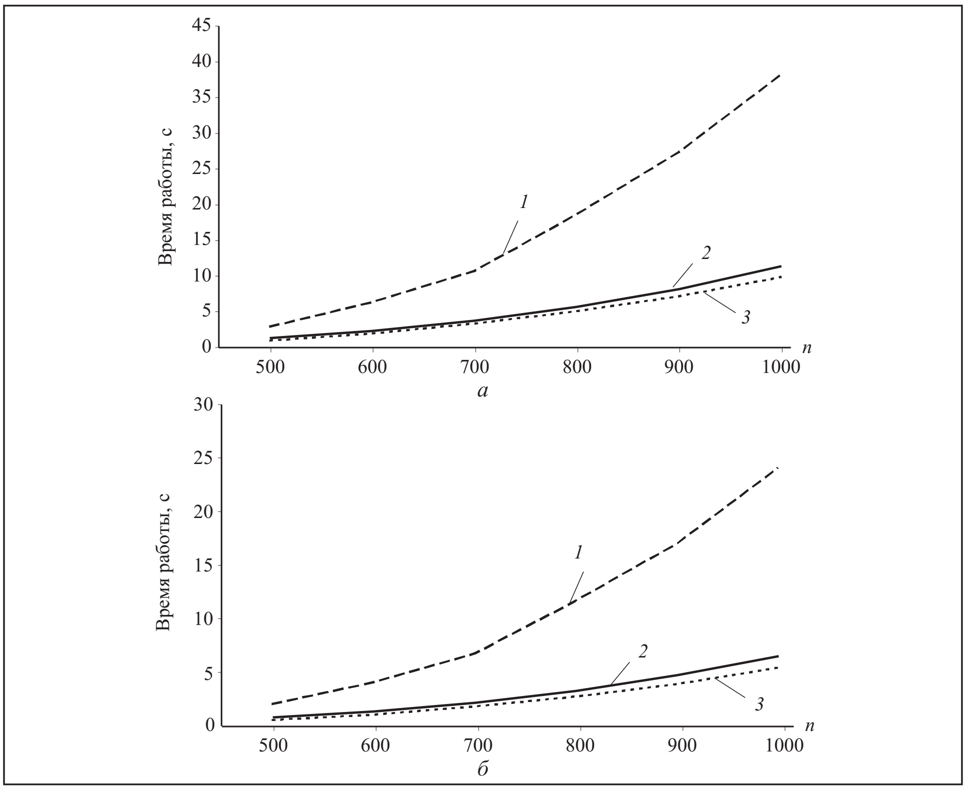

Рис. 3. График изменения времени работы алгоритмов SP1 (*1*), Floyd (*2*), SP2 (*3*) (*а*) и SP1S (*1*), FloydS (*2*) , SP2S (*3*) (*б*) в зависимости от роста числа узлов при $val = 2$

Из графиков на рис. 1 видно, что у алгоритмов SP1 и SP1S вычислительная эффективность лучше, чем у алгоритмов Floyd и FloydS уже при степени узлов $val = 5$, а алгоритмы SP2 и SP2S с ростом степени узлов значительно опережают алгоритмы Floyd и FloydS. Для больших значений параметра val (от 200 до 999) алгоритмы SP1, SP1S оказываются лучше, чем SP2, SP2S. Алгоритмы SP2, SP2S всегда лучше алгоритмов Флойда и с ростом степени узлов быстрее последних в несколько раз.

Графики изменения времени работы алгоритмов SP1, Floyd, SP2 и SP1S, FloydS, SP2S в зависимости от роста числа узлов от 100 до 1000 при $val = 5$ приведены на рис. 2, а при росте числа узлов от 500 до 1000 при $val = 2$ представлены на рис. 3. Для сетей с умеренной разреженностью при росте числа узлов в сети алгоритмы SP1, SP2 и SP1S, SP2S лучше, чем Floyd и FloydS. Для сильно разреженных сетей алгоритмы Floyd и FloydS с ростом числа узлов быстрее алгоритмов SP1 и SP1S, однако хуже, чем алгоритмы SP2 и SP2S.

ЗАКЛЮЧЕНИЕ

Рассмотрена улучшенная версия алгоритма [10] построения всех КП в сети по критериям: минимум дуг в пути; минимум длины пути. Доказано, что время работы алгоритма [10] может быть сокращено за счет применения АТД и уменьшения просмотра узлов во внутренних циклах.

Эмпирически показано, что улучшенный алгоритм на разреженных сетях работает быстрее, чем алгоритм в [10], и при увеличении плотности сети на несколько порядков превышает по скорости работы модифицированный алгоритм Флойда.

В целом эксперимент показал высокую вычислительную эффективность предложенного алгоритма, который может быть включен в состав типового инструментария разработчика и с успехом применяться при решении практических задач нахождения двухкритериальных КП на графах и сетях большой размерности.

СПИСОК ЛИТЕРАТУРЫ

1. Hansen P. Bicriterion path problems // Multiple Criteria Decision Making Theory and Application (Ed. G. Fandel and T. Gal). — Berlin: Springer, 1979. — P. 109–127.
2. Warburton A. Approximation of Pareto optima in multiple-objective, shortest-path problems // Oper. Res. — 1987. — **35**, N 1. — P. 70–79.
3. Ehrgott M., Gandibleux X. Multiple criteria optimization — state of the art annotated bibliographic surveys. — Boston, MA: Kluwer, 2002. — 496 p.
4. Князева Н.А. Использование логических операций для поиска оптимальных путей в сети // II Всесоюз. совещание «Методы и программы решения оптимизационных задач на графах и сетях», 24–26 август 1982 г., Улан-Удэ: Тез. докл. — Новосибирск, 1982. – Ч. 1: Теория, алгоритмы. — С. 85–86.
5. Добролюбов В.В., Педяш В.А. Сетевой алгоритм построения путей с минимальным числом транзитных узлов и с максимальной пропускной способностью // Вычисл. средства в технике и системах связи. — 1979. — № 4. — С. 129–132.
6. Moore E.F. The shortest path through a mace // Proceed. of the Intern. Symp. on the Theory of Switching. — Cambridge Harvard University Press, 1959. — Part II. — P. 285–292.
7. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. — М.: Мир, 1979. — 586 с.
8. Floyd R.W. Algorithm 97: shortest path // Comm. ACM. — 1962. — **5**. — P. 345.
9. Dijkstra F.W. A note on two problems in connection with graphs // Numerical mathematic. — 1959. — N 1. — P. 269–271.
10. Васянин В.А., Савенков А.И. Алгоритм построения кратчайших путей на сети по ступенчатому критерию // Дискрет. и эргатич. системы управления: Сб. науч. тр. — Киев, 1983. — С. 40–49.
11. Bellman R.E. On a routing problem // Quart. Appl. Math. — 1958. — **16**, N 1. — P. 87–90.
12. Shimbel A. Applications of matrix algebra to communication nets // Bulletin of Mathematical Biophysics. — 1951. — **13**. — P. 165–178.

Поступила 30.01.2014