УДК 004.422.636

© **В.А. Васянин**, канд. техн. наук;
Л.П. Ушакова, соискатель

Институт телекоммуникаций и глобального информационного пространства
НАН Украины, г. Киев

СТРУКТУРЫ ДАННЫХ И ПРОЦЕДУРЫ РЕДУКЦИИ МАРШРУТОВ В ЗАДАЧАХ РАСПРЕДЕЛЕНИЯ ПОТОКОВ В КОММУНИКАЦИОННЫХ СЕТЯХ

Рассматриваются абстрактные типы данных для разработки алгоритмов распределения и маршрутизации потоков в коммуникационных сетях. Предложены процедуры редукции, позволяющие значительно сократить требуемые объемы оперативной памяти для представления структур данных при решении задач проектирования новых маршрутов передачи потоков, когда в качестве исходных маршрутов выступает их произвольное комбинаторное множество. Приведены оценки трудоемкости алгоритмов. Рассмотрен пример построения структур данных и проведен численный эксперимент для проверки работоспособности и вычислительной эффективности предложенных алгоритмов.

Ключевые слова: абстрактные типы данных, структуры данных, алгоритмы распределения потоков, редукция транспортных маршрутов.

Введение. Во многих задачах оптимизации распределения и маршрутизации потоков в коммуникационных сетях в качестве входных данных выступают маршруты транспортировки потоков. В терминах теории графов маршрут представляет собой простой путь, заданный перечислением узлов и дуг сети. Допускаются кольцевые маршруты, у которых начальный и конечный узлы совпадают. Множество маршрутов может быть изначально задано или генерироваться по определенным правилам в процессе решения задачи. Поскольку маршруты имеют разную длину по числу входящих в них узлов (или дуг), более короткие маршруты могут совпадать с более длинными маршрутами, как по узлам, так и по дугам. Для распределения потоков в алгоритмах оптимизации используются специальные структуры данных – абстрактные типы данных (АТД) [1, 2], с помощью которых описывается связь между распределяемыми потоками и маршрутами. Процедуры редукции маршрутов позволяют значительно сократить число таких связей в случае, когда количество исходных или сгенерированных маршрутов в комбинаторных задачах оптимизации очень велико.

1. Постановка задачи и алгоритм решения.

Пусть $G(N,P)$ – многопродуктовая сеть с множеством неориентированных топологических дуг P, $p=\left|P\right|$, множеством узлов N, $n=\left|N\right|$, где $\left|\cdot\right|$ – знак мощности множества. Под топологической дугой будем понимать физический отрезок линии связи: железной или автомобильной дороги, кабеля сети передачи данных, телефонного кабеля и т. д., соединяющий два любых узла из множества N так, что между рассматриваемыми узлами на данном отрезке нет больше ни одного узла из N. Узлы сети соответствуют пунктам отправления, получения, сортировки и перегрузки (коммутации) потоков.

На сети задана матрица потоков $U=\left\|u_{ij}\right\|_{n\times n}$, $i,j=\overline{1,n}$. Потоки u_{ij} из источников i в стоки j, $i,j=\overline{1,n}$ должны перевозиться в транспортных средствах или передаваться по каналам связи с заданной периодичностью.

Пусть далее на сети задано множество маршрутов транспортных средств или каналов связи $\{m_k\}$, $k=\overline{1,l}$, каждый из которых состоит из последовательности узлов и топологических дуг сети G, соединяющих начальный и конечный узлы маршрута или канала связи. В конкретной задаче оптимизации множество маршрутов может быть не заданным, а искомым. Для сетей передачи данных маршруты могут быть представлены простыми каналами связи, соединяющими смежные узлы или коммутируемыми каналами связи, соединяющими любую последовательность узлов (выделенными каналами). В частном случае, все заданные маршруты могут совпадать с топологическими дугами сети. Множество $\{m_k\}$ может содержать несколько маршрутов, соединяющих любую пару узлов. Определим также маршрутную мультисеть $G_M(N,P_M)$, где N – множество узлов сети, P_M – множество ее ориентированных маршрутных дуг. Между любыми узлами i и j сети G_M существует маршрутная дуга, если они связаны хотя бы одним маршрутом транспортного средства или каналом связи из $\{m_k\}$.

Как известно, вычислительная эффективность алгоритмов во многом зависит от удачного представления структур данных, используемых в процессе решения задачи. При решении задач распределения и маршрутизации потоков необходимо иметь такую структуру данных, которая, с одной стороны, позволяла бы для каждого потока u_{ij} быстро находить множество маршрутов, на которые может быть распределен данный поток, а также определять множество потоков, которые могут быть реализованы на данном маршруте. С другой стороны, структура данных должна обеспечивать хранение результатов решения задачи – схему распределения потоков, в которой для каждого u_{ij} определен путь следования от начального i-го узла до конечного j-го узла с указанием маршрутов и транзитных узлов. Другим важным требованием, предъявляемым к организации структуры данных, является минимум информации, хранимой в процессе решения задачи.

Пусть $\upsilon_k=\{\xi_1,\xi_2,...,\xi_{\eta_k}\}$ – упорядоченное множество узлов из N на маршруте m_k, $\left|\upsilon_k\right|=\eta_k$, $q_k=\{(\xi_1,\xi_2),(\xi_2,\xi_3),...,(\xi_{\eta_k-1},\xi_{\eta_k})\}$, $\left|q_k\right|=\eta_k-1$ – упорядоченное множество дуг из P, составляющих маршрут m_k.

Поскольку в большинстве случаев прямые и обратные маршруты, как правило, совпадают, в целях экономии оперативной памяти ЭВМ, для совпадающих маршрутов представляется разумным хранить только прямые маршруты, подразумевая, что транспортировка потока по такому маршруту разрешена в обе стороны. Если прямой и обратный маршруты не совпадают, то каждый из них должен быть представлен отдельно.

Для различия маршрутов будем задавать признак направления движения по маршруту. Пусть $\{m_k\}$, $k=\overline{1,l}$ – множество маршрутов, полученное с учетом сделанных замечаний, а MV_k – признак направления движения по маршруту, где

$$MV_k = \begin{cases} 0, \text{если движение разрешено в обоих направлениях,} \\ 1, \text{если движение разрешено в одну сторону.} \end{cases}$$

Введем также дополнительные признаки маршрутов: $MT_k = \lambda$, где λ – целое число, определяющее тип маршрута;

$$MZ_k = \begin{cases} 0, \text{если движение по маршруту разрешено,} \\ 1, \text{если движение запрещено.} \end{cases}$$

Дополнительные признаки будут использоваться в алгоритмах редукции маршрутов и формирования справочной структуры данных.

Определим справочную структуру H_S, состоящую из матрицы указателей $\Lambda = \| \Delta_{ij} \|_{n \times n}$ и элементов E, связанных в однонаправленные линейные списки.

Указатель Δ_{ij} соответствует потоку u_{ij} и указывает на первый элемент E, называемый головой списка. Каждый элемент E может иметь ссылку на следующий элемент. Последний элемент в списке имеет нулевую ссылку ($NULL$ - ссылку) и называется хвостовым. Каждый элемент E состоит из поля FWD, содержащего ссылку на следующий элемент или значение $NULL$; поля AM, указывающего адрес маршрута, на который может быть распределен данный поток и поля признаков DM. Поле признаков состоит из пяти подполей, которые принимают следующие значения:

$$DM_1 = \begin{cases} 1, \text{если поток } u_{ij}, i < j \text{ выбран на маршрут } AM, \\ 0, \text{в противном случае;} \end{cases}$$

$$DM_2 = \begin{cases} 1, \text{если поток } u_{ij}, i > j \text{ выбран на маршрут } AM, \\ 0, \text{в противном случае;} \end{cases}$$

$$DM_3 = \begin{cases} 1, \text{если для } u_{ij} \text{ выполняется } (i < j) \wedge (i \prec j), \\ 0, \text{если для } u_{ij} \text{ выполняется } (i < j) \wedge (i \succ j); \end{cases}$$

$$DM_4 = \begin{cases} 1, \text{если для } u_{ij} \text{ выполняется } (i > j) \wedge (i \prec j), \\ 0, \text{если для } u_{ij} \text{ выполняется } (i > j) \wedge (i \succ j); \end{cases}$$

$$DM_5 = \begin{cases} 0, \text{если достигнут конец цепочки элементов } E, \text{определяющих маршруты,} \\ \text{на которых поток } u_{ij} \text{ может быть распределен без перегрузок} \\ 1, \text{в противном случае.} \end{cases}$$

Знаки $i \prec j$, $i \succ j$ соответственно означают, что узел i предшествует узлу j или следует за узлом j при движении по маршруту AM в порядке прохождения узлов из υ_{AM}, а '$\wedge$' – операция конъюнкции (логического «и»).

Поток u_{ij} назовем безтранзитным, если для него выполняются условия: $\exists m_k \in \{m_k\}$, $k = \overline{1,l}$, для которого справедливо $(i, j \in \upsilon_k \wedge MV_k = 0) \vee (i, j \in \upsilon_k \wedge MV_k \neq 0 \wedge i \prec j)$, где '$\exists$' – знак существования, '$\vee$' – операция дизъюнкции (логического «или»). В противном случае, поток будем называть транзитным. При начальном формировании структуры H_S, количество элементов E в списке для безтранзитного потока u_{ij} определяется числом маршрутов, на которые может быть распределен поток, причем, если симметричный поток u_{ji} также безтранзитный, то соответствующие указатели Δ_{ij} и Δ_{ji} получают ссылки на один список. Для транзитных потоков, которые должны быть один или более раз перегружены с одного маршрута на другой, соответствующие указатели из Λ первоначально получают нулевые ссылки. Построение цепочек элементов E для транзитных потоков u_{ij} осуществляется в процессе решения задачи и заключается в следующем: определяются маршруты, на которые может быть распределен поток; для найденных маршрутов создаются элементы E, которые связываются в список в соответствии с порядком следования по маршрутам от i до j. В справочной H_S для каждого транзитного потока запоминается только один список, определяющий путь, который был выбран для распределения данного потока в процессе оптимизации, промежуточные списки не запоминаются. Если в процессе решения задачи безтранзитный поток переводится в ранг транзитных, то выбранный для него список связывается с первоначальным. Для разграничения списков в этом случае используется подполе DM_5. Для транзитных потоков каждый элемент E содержит дополнительное поле AY, в которое записывается узел маршрута AM, в котором перегружается поток. Последний элемент E в цепочке для транзитного потока u_{ij} в поле AY будет содержать узел j. В случае, когда при решении задачи разрешено разветвление (дробление) потоков, в структуру элементов E могут быть введены дополнительные поля для хранения части распределенного потока.

На рис. 1 показан пример структуры H_S, поясняющий введенные понятия.

Из правил начального формирования H_S ясно, что число элементов E в списках H_S зависит от числа маршрутов и определяется следующим образом:

$$E_{HS} = \sum_{k=1}^{l} \eta_k (\eta_k - 1)/2.$$

Величина E_{HS} для реальных сетей может быть весьма значительной. Так, например, для сети, содержащей 300 узлов, при генерации для всех симметричных $n(n-1)/2$ потоков кратчайших путей распределения потоков (отдельных маршрутов) будет сгенерировано 44850 маршрутов. При средней длине маршрута η_{cp} = 15 в структуру H_S будет включено $44850\eta_{cp}(\eta_{cp}-1)/2 = 44850\times 105 = 4709250$ элементов, для размещения которых потребуется

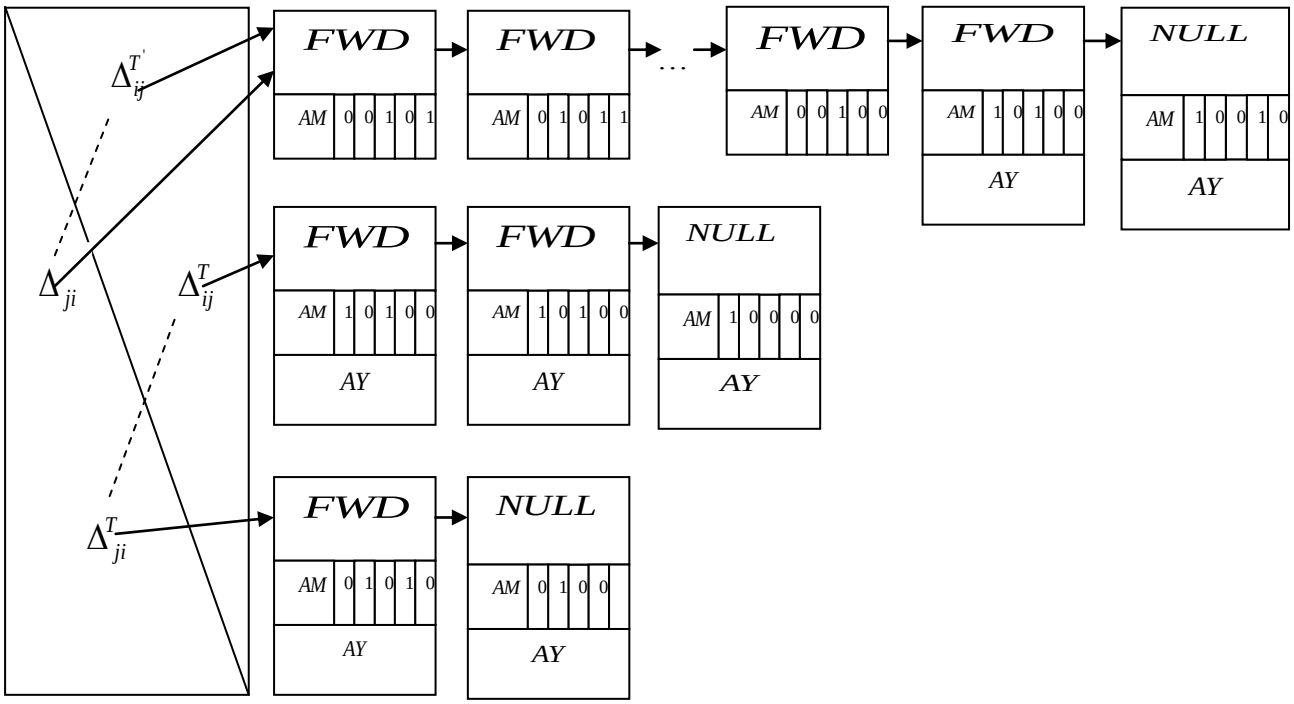


Рис. 1 – Где: $\Delta_{ij}^{T'}$ – безтранзитный поток, переведенный в ранг транзитных; Δ_{ji} – безтранзитный поток; Δ_{ij}^{T}, Δ_{ji}^{T} – симметричные транзитные потоки

порядка 45 Мб оперативной памяти (учитывая, что для размещения одного элемента E требуется 10 байтов).

В связи с этим, рассмотрим процедуры редукции, позволяющие сократить размерность структуры H_S. Пусть маршрут m_α редуцируется в маршрут m_β по узлам, если выполняется условие:

$$\upsilon_\alpha \cap \upsilon_\beta = \upsilon_\alpha,$$

и редуцируется по топологическим дугам, если

$$q_\alpha \cap q_\beta = q_\alpha.$$

Очевидно, что редукция маршрутов сокращает количество связей в сети G_M и число элементов в структуре H_S на величину:

$$\sum_{\alpha\in\{k^*\}}\eta_\alpha(\eta_\alpha-1)/2,$$

где $\{k^*\}$ – множество номеров редуцированных маршрутов.

Пусть далее LM_i, $i=\overline{1,l}$ – вектор длин маршрутов по числу узлов, VM_i, $i=\overline{1,l}$ – вектор номеров маршрутов, упорядоченных по убыванию числа узлов в маршруте. В процессе редукции маршрутов будем строить вектор редукции VR_i, $i=\overline{1,l}$ и справочный вектор RV_i, $i=\overline{1,l}$. Элементы векторов определим следующим образом:

$$VR_i=\begin{cases}0, \text{ если } i\text{- ый маршрут не редуцирован ни в какой другой маршрут,}\\ k, \text{ если } i\text{- ый маршрут редуцирован в некоторый маршрут } \xi.\end{cases}$$

Где k – номер маршрута, который был редуцирован в маршрут ξ непосредственно перед маршрутом i. Если i – первый редуцируемый маршрут, то $k=\xi$.

$$RV_i=\begin{cases}0, \text{ если } i\text{- ый маршрут редуцирован,}\\ \xi, \text{ если в маршруте } i \text{ редуцирован какой-либо маршрут,}\\ \text{но сам он не редуцирован ни в какой другой маршрут,}\\ i, \text{ если маршрут } i \text{ сам не редуцирован ни в какой другой}\\ \text{и в него не редуцирован ни один маршрут.}\end{cases}$$

Где ξ – номер последнего маршрута, редуцированного в маршрут i.

Каждый элемент вектора $VR_i=0$, соответствующий маршруту m_i определяет цепочку редуцированных в этот маршрут других маршрутов. Упорядочение маршрутов в цепочке обратно порядку их редукции. На первый маршрут в цепочке для VR_i указывает элемент RV_i.

Приведем алгоритм редукции. Пусть $S=\left\| s_{ij} \right\|_{l\times\eta_{\max}}$ – матрица маршрутов, заданная перечислением номеров узлов сети, где $\eta_{\max}$ – максимальное число узлов в маршруте. Предполагается, что номера недостающих узлов в более коротких маршрутах заменены на 0.

Обозначим через $BR1$, $BR2$ – рабочие векторы размерностью n. Для большей наглядности в записи псевдокода алгоритма для индексов матриц и векторов будем использовать круглые скобки. Знак '$\leftarrow$' означает операцию присваивания, а знаком '$***$' отделяются комментарии.

Алгоритм 1. Редукция маршрутов.

1. Для $\{i \mid i=\overline{1,l}\}$ выполнить $RV(i)\leftarrow i$, $VR(i)\leftarrow 0$.
2. Если выбрана однородная редукция, то $RT\leftarrow'O'$, для смешанной редукции $RT\leftarrow'M'$. *** Для однородной редукции типы маршрутов $MT_k=\lambda$ совпадают
3. Если выбрана редукция по узлам, то $PRT\leftarrow'U'$, для редукции по дугам $PRT\leftarrow'A'$.
4. Если $PRT='U'$, то перейти к 5, иначе перейти к 25.
5. Для $\{i \mid i=\overline{1,l-1}\}$ выполнить 6–24. *** Начало цикла по i, редукция по узлам

6. $K1 \leftarrow VM(i)$.
7. Если $VR(VM(i)) = 0$, то перейти к 8, иначе к 24.
8. $BR1 \leftarrow 0$.
9. Для $\{k \mid k = \overline{1, LM(VM(i))}\}$ выполнить $BR1(S(VM(i), k)) \leftarrow 1$.
10. Для $\{j \mid j = \overline{i+1, l}\}$ выполнить 11–23. *** Начало цикла по j
11. Если $VR(VM(j)) = 0$, то перейти к 12, иначе к 23.
12. $BR2 \leftarrow 0$.
13. Для $\{k \mid k = \overline{1, LM(VM(j))}\}$ выполнить $BR2(S(VM(j), k)) \leftarrow 1$.
14. Если $RT = 'M' \vee RT = 'O' \wedge MT(VM(i)) = MT(VM(j))$, то перейти к 15, иначе к 23.
15. Для $\{k \mid k = \overline{1, n}\}$ выполнить 16–18. *** Начало цикла по k
16. Если $BR2(k) = 1$, то перейти к 17, иначе к 18.
17. Если $BR1(k) = 1$, то перейти к 18, иначе к 23.
18. Перейти к 15. *** Конец цикла по k
19. $RV(VM(j)) \leftarrow 0$.
20. $RV(VM(i)) \leftarrow VM(j)$.
21. $VR(VM(j)) \leftarrow K1$.
22. $K1 \leftarrow VM(j)$.
23. Перейти к 10. *** Конец цикла по j
24. Перейти к 5. *** Конец цикла по i
25. Для $\{i \mid i = \overline{1, l-1}\}$ выполнить 26–55. *** Начало цикла по i, редукция по дугам
26. $KS \leftarrow VM(i)$.
27. Если $VR(VM(i)) \neq 0$, то перейти к 55, иначе к 28.
28. Для $\{j \mid j = \overline{i+1, l}\}$ выполнить 29–54. *** Начало цикла по j
29. Если $VR(VM(j)) \neq 0$, то перейти к 54, иначе к 30.
30. $KL \leftarrow 0$; $KF \leftarrow 1$.
31. Для $\{k1 \mid k1 = \overline{1, LM(VM(i))}\}$ выполнить 32–34.
32. Если $S(VM(j), KF) = S(VM(i), k1)$, то перейти к 36.
33. Если $S(VM(j), LM(VM(j)) = S(VM(i), k1)$, то перейти к 48.
34. Перейти к 31.
35. Перейти к 54, продолжить цикл по j.
36. Для $\{k \mid k = \overline{1, LM(VM(j))}\}$ выполнить 37–40.
37. Если $S(VM(j), k) = S(VM(i), k1)$, то перейти к 38, иначе к 40.
38. $KL \leftarrow KL + 1$; $k1 \leftarrow k1 + 1$.
39. Если $k1 > LM(VM(i))$, то перейти к 41.
40. Перейти к 36.
41. Если $RT = 'M' \vee RT = 'O' \wedge MT(VM(i)) = MT(VM(j))$, то перейти к 42, иначе к 47.
42. Если $KL = LM(VM(j))$, то перейти к 43, иначе к 47.
43. $RV(VM(j)) \leftarrow 0$.
44. $RV(VM(i)) \leftarrow VM(j)$.
45. $VR(VM(j)) \leftarrow KS$.

46. $KS \leftarrow VM(j)$.
47. Перейти к 54, продолжить цикл по j.
48. Для $\{k \mid k = \overline{LM(VM(j)),1,-1}\}$ выполнить 49–52.
49. Если $S(VM(j),k) = S(VM(i),k1)$, то перейти к 50, иначе к 52.
50. $KL \leftarrow KL+1;\ k1 \leftarrow k1+1$.
51. Если $k1 > LM(VM(i))$, то перейти к 41.
52. Перейти к 48.
53. Перейти к 41.
54. Перейти к 28. *** Конец цикла по j
55. Перейти к 25. *** Конец цикла по i
56. Стоп.

В результате работы приведенного алгоритма цепочки в векторе редукции для $VR_i = 0$, на начало которых указывает RV_i, будут упорядочены по возрастанию числа узлов в редуцированных маршрутах.

После выполнения процедуры редукции строится справочная структура. При этом просматриваются только такие маршруты, у которых $VR_i = 0$. Очевидно, что после выполнения процедуры редукции объем оперативной памяти ЭВМ, занимаемой справочной H_S, сократится на величину:

$$\Delta = \left(\sum_{k=1}^{l} \eta_k (\eta_k - 1)/2 - \sum_{\alpha \in \{k^*\}} \eta_\alpha (\eta_\alpha - 1)/2 \right) V_E,$$

где V_E – объем памяти, занимаемый одним элементом списка.

Из алгоритма видно, что трудоемкость редукции по узлам составляет в среднем $O(l^2(n+\eta_{cp})+l\eta_{cp})$, а по дугам – $O(l^2\eta_{cp})$ операций, где η_{cp} – средняя длина маршрута.

Необходимо отметить, что редукция маршрутов несколько увеличит трудоемкость основного алгоритма оптимизации, так как возрастет число просматриваемых маршрутов в шагах выбора маршрутов для распределения потоков. Несмотря на это, редукция маршрутов оказывается крайне необходимой, если в качестве исходного множества маршрутов выступают все возможные маршруты [3]. В этом случае резко возрастает размерность задачи, принимающей характер построения новых, ранее не существующих маршрутов.

Рассмотрим алгоритм начального формирования справочной H_S с одновременным построением маршрутных дуг сети G_M.

Пусть $R = \| r_{ij} \|_{n \times n}$ – топологическая матрица сети G, r_{ij} – длина дуги p_{ij} (если дуги p_{ij} не существует, $r_{ij} = 0$); $D = \| d_{ij} \|_{n \times n}$ – матрица длин маршрутных дуг (первоначально $d_{ij} = \infty$, $i, j = \overline{1,n}$).

Поскольку любые два узла могут быть связаны более чем одним маршрутом, длины дуг d_{ij} будем определять из следующего выражения:

$$d_{ij} = \min_{\{v_k\}}(r_{\xi_\mu,\xi_{\mu+1}} + r_{\xi_{\mu+1},\xi_{\mu+2}} + ... + r_{\xi_{\mu+c-1},\xi_{\mu+c}}),\ \text{при}\ \xi_\mu = i,\ \xi_{\mu+c} = j,\ k = \overline{1,l}.$$

В описании алгоритма запись «Образовать новый элемент $E(P)$» означает генерацию нового элемента E и установку указателя P на адрес элемента. Обращение к полям элемента будем записывать в виде $P.\langle\text{наименование поля}\rangle$, а установку указателя в виде $P \Rightarrow$.

Алгоритм 2. Формирование H_S.

1. Для $\{i \mid i = \overline{l,1,-1}\}$ выполнить 2–18. *** Начало цикла по i, формирование H_S
2. Если $VR(VM(i)) = 0 \wedge MZ(i) = 0$, то перейти к 3, иначе к 18.
3. Для $\{k \mid k = \overline{1, LM(VM(i)) - 1}\}$ выполнить 4–17. *** Начало цикла по k
4. Для $\{j \mid j = \overline{k+1, LM(VM(i))}\}$ выполнить 5–16. *** Начало цикла по j
5. Образовать новый элемент $E(P)$.
6. Если $\Lambda(S(VM(i),k), S(VM(i), j)).FWD \neq NULL$, то перейти к 7, иначе к 12.
7. $P1 \Rightarrow \Lambda(S(VM(i),k), S(VM(i), j)).FWD$.
8. Пока $P1.FWD \neq NULL$, выполнить 9–10.
9. $P1 \Rightarrow P1.FWD$.
10. Перейти к 8.
11. $P1.FWD \Rightarrow P$. Перейти к 14.
12. $\Lambda(S(VM(i),k), S(VM(i), j)).FWD \Rightarrow P$.
13. $\Lambda(S(VM(i), j), S(VM(i),k)).FWD \Rightarrow P$. Перейти к 14.
14. $P.FWD \Rightarrow NULL$; $P.AM \leftarrow VM(i)$; $P.DM \leftarrow 0$; $P.DM(5) \leftarrow 1$.
15. Если $S(VM(i),k) < S(VM(i), j))$, то $P.DM(3) \leftarrow 1$, иначе $P.DM(4) \leftarrow 1$.
16. Перейти на 4. *** Конец цикла по j
17. Перейти на 3. *** Конец цикла по k
18. Перейти на 1. *** Конец цикла по i
19. $D \leftarrow \infty$.
20. Для $\{i \mid i = \overline{1,l}\}$ выполнить 21–30. *** Начало цикла по i, формирование D
21. Для $\{k \mid k = \overline{1, LM(i) - 1}\}$ выполнить 22–29. *** Начало цикла по k
22. $SUM \leftarrow 0$.
23. Для $\{j \mid j = \overline{k+1, LM(i)}\}$ выполнить 24–28. *** Начало цикла по j
24. $m \leftarrow j - 1$.
25. $SUM \leftarrow SUM + R(S(i,m), S(i, j))$.
26. Если $D(S(i,k), S(i, j)) > SUM$, то $D(S(i,k), S(i, j)) = SUM$.
27. Если $MV(i) = 0 \wedge D(S(i, j), S(i,k)) > SUM$, то $D(S(i, j), S(i,k)) = SUM$.
28. Перейти к 23. *** Конец цикла по j
29. Перейти к 21. *** Конец цикла по k
30. Перейти к 20. *** Конец цикла по i
31. Стоп.

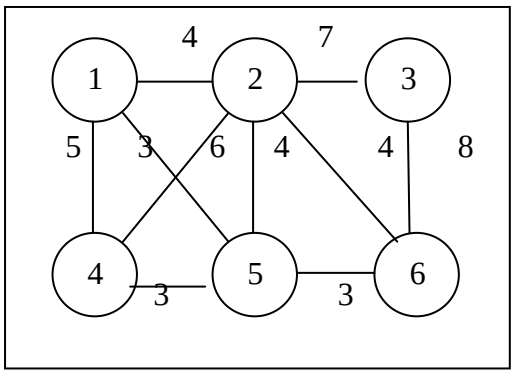
Как нетрудно видеть, средняя трудоемкость алгоритма 2 составляет $O((l-lr)\eta_{cp}^2)$ операций, где lr – число редуцированных маршрутов, а η_{cp} – средняя длина маршрута.

Для начального распределения потоков на сети G_M используется алгоритм построения кратчайших путей по критерию: минимум транзитных перегрузок; при равенстве числа перегрузок – минимум длины пути [4].

2. Пример формирования справочной структуры H_S.

Пусть задана сеть с числом узлов $n=6$, числом дуг $p=10$ и матрицей длин дуг R, приведенная на рис. 2. После редукции заданных маршрутов по узлам (U) и дугам (A) были получены результаты, показанные в табл. 1 и на рис. 3.

На рис. 4 приведены фрагменты справочной структуры H_S для потоков u_{12} и u_{21} без редукции маршрутов и с редукцией по узлам и дугам. При распределении потоков в первом случае маршруты будут просматриваться в следующем порядке – 8, 11, 5, 2, 4, 3, 1; во втором – 6, 13, 8, 7, 11, 10, 5, 14, 12, 2, 4, 3, 9, 1; в третьем – 8, 6, 5, 4, 10, 3, 1, 13, 11, 2, 1. Таким образом, как уже отмечалось, после редукции маршрутов в основном алгоритме оптимизации общее число просмотров маршрутов увеличивается, что соответственно приводит к увеличению его трудоемкости. Кроме того, в случае редукции по дугам просмотр маршрутов по возрастанию числа узлов может быть нарушен, если какой-либо один и тот же поток может проходить по путям с разными дугами. В нашем примере, это оба потока – u_{12} и u_{21} (маршруты 1, 2, 3, 4, 5, 11 и маршрут 8).



	R					
N	1	2	3	4	5	6
1	**0**	**4**	**0**	**5**	**3**	**0**
2	**4**	**0**	**7**	**6**	**4**	**4**
3	**0**	**7**	**0**	**0**	**0**	**8**
4	**5**	**6**	**0**	**0**	**3**	**0**
5	**3**	**4**	**0**	**3**	**0**	**3**
6	**0**	**4**	**8**	**0**	**3**	**0**

Рис. 2

Таблица 1

	Редукция	
	по узлам U	по дугам A
Число узлов в сети	6	6
Число дуг в сети	10	10
Число маршрутов в сети	14	14
Число редуцированных маршрутов, в т. ч. в %	13, 92%	6, 42%
Число элементов в структуре АТД	72	72
Число редуцированных элементов, в т. ч. в %	57, 79%	17, 23%

№	$S = \| s_{ij} \|_{14\times 6}$						LM_{14}	VM_{14}	VR_{14}	RV_{14}	VR_{14}	RV_{14}
									U		A	
1	1	2	3	6	5	4	6	1	0	6	0	13
2	1	2	3	6	0	0	4	9	4	0	1	0
3	1	2	6	5	0	0	4	3	9	0	0	3
4	1	2	5	6	0	0	4	4	3	0	0	10
5	1	2	4	0	0	0	3	2	14	0	0	5
6	1	4	0	0	0	0	2	12	13	0	8	0
7	2	4	5	0	0	0	3	14	11	0	9	0
8	2	4	1	0	0	0	3	5	7	0	0	6
9	2	4	5	6	3	0	5	10	1	0	0	7
10	2	5	6	0	0	0	3	11	5	0	4	0
11	3	2	1	0	0	0	3	7	10	0	2	0
12	3	2	5	6	0	0	4	8	2	0	0	12
13	3	6	0	0	0	0	2	13	8	0	11	0
14	4	5	2	6	0	0	4	6	12	0	0	14

Рис. 3

На рис. 5 показана матрица длин кратчайших путей D и справочная матрица построенных путей $C = \| c_{ij} \|_{n\times n}$, каждый элемент которой c_{ij}, $i \neq j$ определяет номер предпоследнего узла на кратчайшем пути от i до j, $c_{ii} = 0$, $i = \overline{1,n}$. Матрицы получены для сети G_M алгоритмом [4].

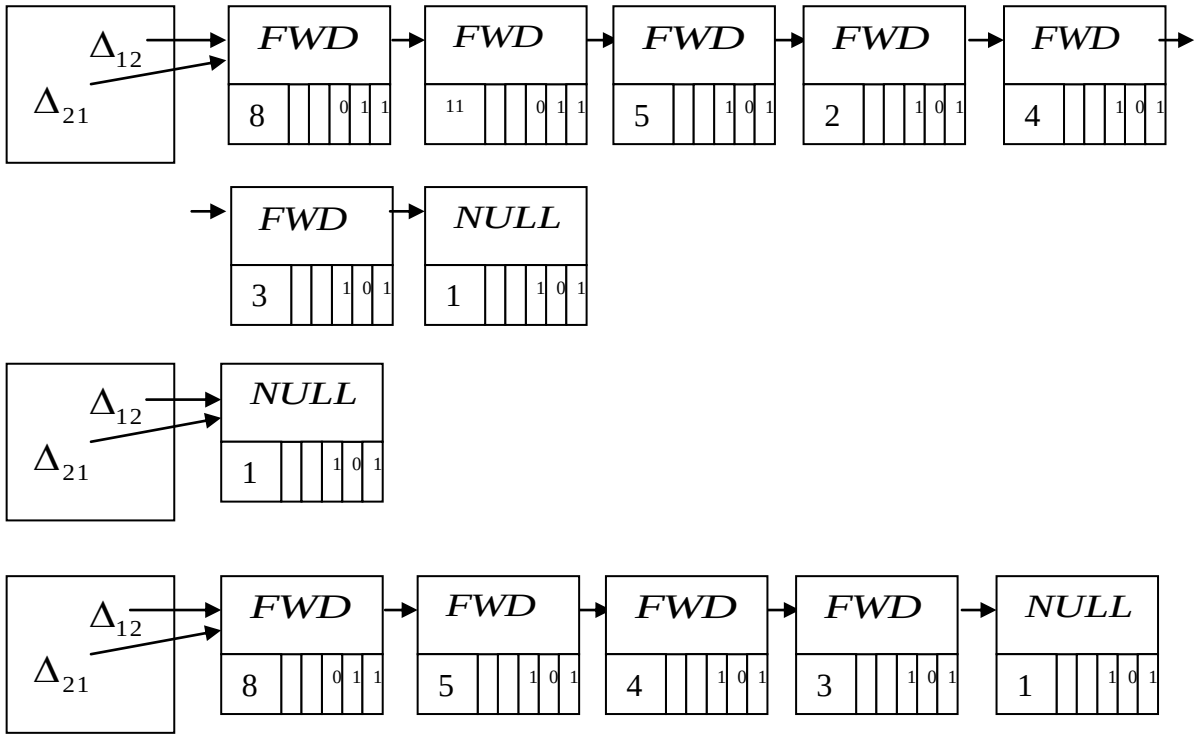


Рис. 4

D						
N	1	2	3	4	5	6
1	**0**	**4**	**11**	**5**	**8**	**8**
2	**4**	**0**	**7**	**6**	**4**	**4**
3	**11**	**7**	**0**	**14**	**11**	**8**
4	**5**	**6**	**14**	**0**	**3**	**6**
5	**8**	**4**	**11**	**3**	**0**	**3**
6	**8**	**4**	**8**	**6**	**3**	**0**

C						
N	1	2	3	4	5	6
1	**0**	**1**	**1**	**1**	**1**	**1**
2	**2**	**0**	**2**	**2**	**2**	**2**
3	**3**	**3**	**0**	**3**	**3**	**3**
4	**4**	**4**	**4**	**0**	**4**	**4**
5	**5**	**5**	**5**	**5**	**0**	**5**
6	**6**	**6**	**6**	**6**	**6**	**0**

Рис. 5

3. Результаты вычислительного эксперимента.

Для проверки работоспособности и вычислительной эффективности алгоритмов был проведен вычислительный эксперимент на ПЭВМ IP-IV с тактовой частотой 2,66 ГГц и оперативной памятью 2 Гб под управлением операционной системы Windows Vista. Проверка алгоритмов осуществлялась на сетях, содержащих от 50 до 500 узлов. Для всех размерностей датчиком псевдослучайных чисел генерировались однородные сети с числом исходящих дуг из каждого узла (степенью узлов) $val = 5$ и длинами топологических дуг в пределах от 80 до 320. Результаты расчетов приведены в табл. 2 и на рисунках 6–9. Где lr и $lr\%$, E, Er и $Er\%$, $TFLOYD$, $TRED,U$, $TRED,A$, $TFORM$, $TVVSP$ – соответственно число и процент редуцированных маршрутов; число сгенерированных и число и процент редуцированных элементов E в структуре H_S; время построения кратчайших путей по критерию – минимум длины пути на сети G алгоритмом Флойда для формирования маршрутов для всех $n^2 - n$ потоков; время редукции маршрутов по узлам и дугам; время формирования структуры H_S; время построения кратчайших путей по критерию – минимум транзитных перегрузок, при равенстве числа перегрузок – минимум длины пути на сети G_M алгоритмом [4].

Таблица 2

n	50	100	150	200	250	300	400	500
val	5	5	5	5	5	5	5	5
p	125	250	375	500	625	750	1000	1250
l	1225	4950	11175	19900	31125	44850	79800	124750
$\eta_{\max}$	7	9	10	13	11	12	13	14
η_{cp}	3	4	5	5	6	6	6	6
lr $lr\%$	789, 64%	3382, 68%	7764, 69%	14165, 71%	22102, 71%	31336, 69%	55682, 69%	88619, 71%
E	7797	48961	138596	301079	510913	786289	1618257	2822650
Er $Er\%$	3904, 50%	27196, 55%	79606, 57%	181025, 60%	308204, 60%	467208, 59%	967801, 59%	1743565, 61%
$TFLOYD$	0,02	0,02	0,05	0,08	0,16	0,28	0,64	1,20
$TRED,U$ $TRED,A$	0,02 0,00	0,28 0,16	2,12 0,61	6,32 1,81	17,88 4,35	53,38 9,47	184,81 31,47	546,85 85,47
$TFORM$	0,00	0,02	0,03	0,06	0,11	0,27	1,11	2,65
$TVVSP$	0,00	0,00	0,00	0,00	0,00	0,00	0,02	0,00

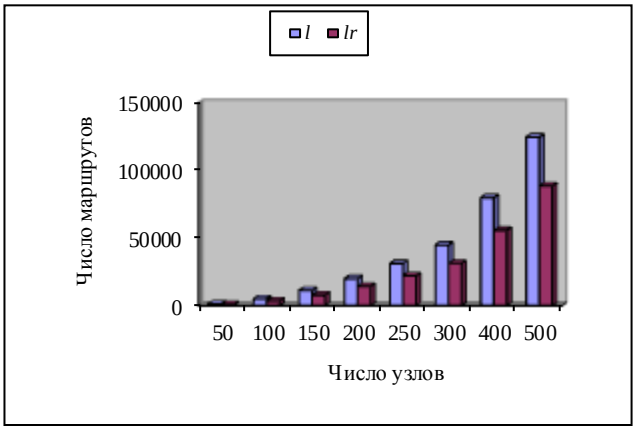


Рис. 6

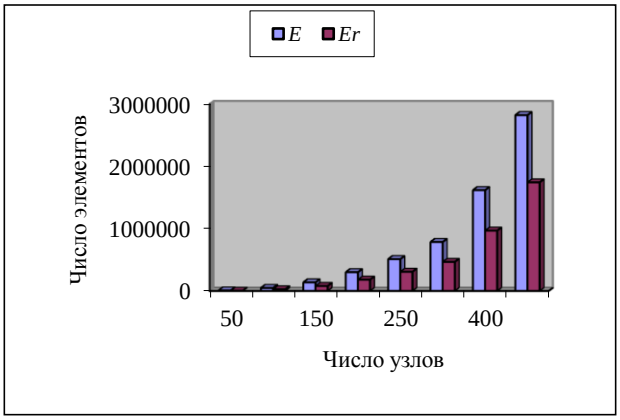


Рис. 7

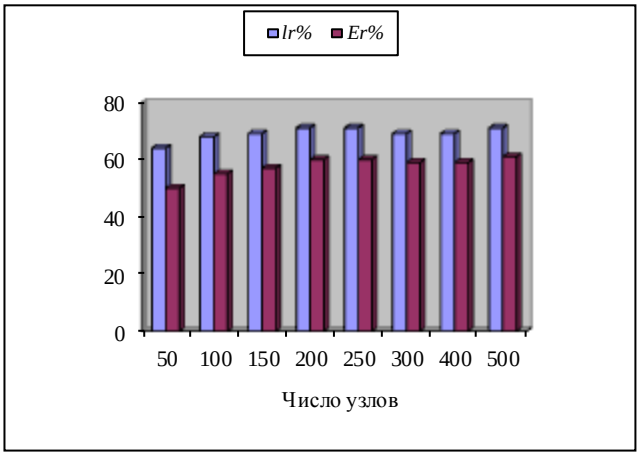


Рис. 8

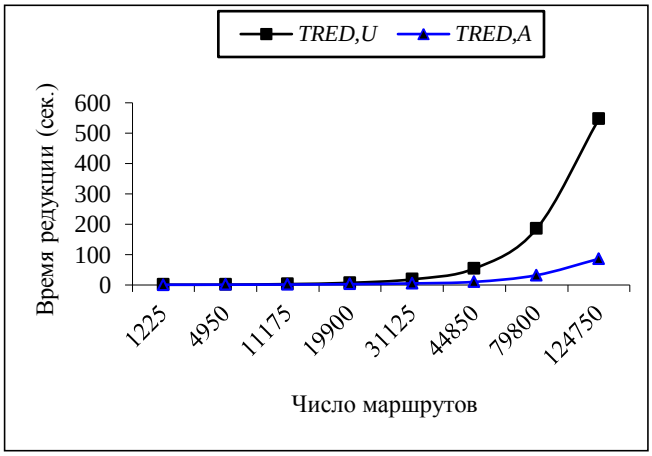


Рис. 9

Выводы. 1. Предложены эффективные структуры данных, основанные на линейных динамических списках, с помощью которых можно быстро найти все возможные маршруты для распределения потоков в многопродуктовой сети. Разработаны алгоритмы редукции маршрутов, которые позволяют решать задачи распределения потоков в случае, когда число рассматриваемых маршрутов очень велико, а объем оперативной памяти ЭВМ ограничен.

2. Получены оценки трудоемкости алгоритмов редукции маршрутов по узлам – $O(l^2(n+\eta_{cp})+l\eta_{cp})$ и дугам – $O(l^2\eta_{cp})$, а также алгоритма формирования справочной структуры данных – $O((l-lr)\eta_{cp}^2)$, где l и n – число маршрутов и узлов в сети, η_{cp} – средняя длина маршрута, lr – число редуцированных маршрутов. Проведено тестирование предложенных алгоритмов на сетях с числом узлов от 50 до 500 и числом маршрутов от 1225 до 124750, которое показало их работоспособность и хорошую вычислительную эффективность и они могут быть использованы в практических задачах распределения и маршрутизации потоков на сетях большой размерности.

Список использованной литературы

1. Седжвик Р. Фундаментальные алгоритмы на C++. Алгоритмы на графах: Пер. с англ. / Р. Седжвик. – СПб: ООО «ДиаСофтЮП», 2002. – 496 с.

2. Кормен Т. Алгоритмы: построение и анализ, 2-е издание / Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн: Пер. с англ. – М.: Издательский дом "Вильямс", 2005. – 1296 с.

3. Васянин В.А. Субоптимальный алгоритм решения задачи маршрутизации перевозок / В.А. Васянин, А.И. Савенков // Автоматика. – Киев, 1982. – № 6. – С. 5–9.

4. Васянин В.А. Алгоритм построения кратчайших путей на сети по ступенчатому критерию / В.А. Васянин, А.И. Савенков // Дискретные и эргатические системы управления: Сб. науч. тр. – Киев, 1983. – С. 40–49. – В надзаг.: АН УССР, Научный совет по проблеме "Кибернетика", институт кибернетики.

Стаття надійшла до редакції 22.01.14 російською мовою

© В.О. Васянін, Л.П. Ушакова

СТРУКТУРИ ДАНИХ ТА ПРОЦЕДУРИ РЕДУКЦІЇ МАРШРУТІВ В ЗАДАЧАХ РОЗПОДІЛУ ПОТОКІВ В КОМУНІКАЦІЙНИХ МЕРЕЖАХ

Обговорюються абстрактні типи даних для розробки алгоритмів розподілу та маршрутизації потоків у комунікаційних мережах. Запропоновані процедури редукції, дозволяючи значно скоротити потрібний обсяг оперативної пам'яті для представлення структур даних при розв'язанні задач проектування нових маршрутів передачі потоків, коли як джерело маршрутів виступає їх довільна комбінаторна множина. Наведені оцінки складності алгоритмів. Розглядається приклад побудови структури даних та проведено чисельний експеримент для перевірки функціональності та ефективності запропонованих обчислювальних алгоритмів.

© V.A. Vasyanin, L.P. Ushakova

STRUCTURES OF DATA AND PROCEDURES OF REDUCTION OF ROUTES IN PROBLEMS OF FLOWS DISTRIBUTION IN THE COMMUNICATIONS NETWORKS

The abstract types of data for developing of algorithms of routing of flows in the communications networks are examined. Are proposed the procedures of reduction, which make it possible to considerably reduce of required working storage for the structures of data with the solution of the problems of designing the new routes of the transfer of the flows, when the set of the initial routes comes out of arbitrary combinatory set. The estimations of the complexity of algorithm are given. An example of the construction of the structures of data is examined and is carried out the computational experiment for test the functionality and effectiveness of the algorithms proposed.