УДК 519.163

СПРАВОЧНАЯ МАТРИЦА СЛИЯНИЯ ПОТОКОВ В ЗАДАЧАХ ОПТИМИЗАЦИИ УПАКОВОК НА МНОГОПРОДУКТОВЫХ СЕТЯХ

В.А. ВАСЯНИН

Предложен способ формирования справочной матрицы слияния мелкопартионных дискретных потоков при решении задач оптимизации упаковок в многопродуктовых сетях с ограничениями на время доставки или число слияний потоков. На основании доказанных утверждений разработаны эффективные вычислительные алгоритмы для определения узлов слияния и слитых потоков с помощью справочной матрицы для всех корреспондирующихся пар в сети. Алгоритмы могут быть использованы внутри основных схем оптимизации для расчета времени доставки потоков адресату и проверки соответствующих ограничений при решении различных задач оптимизации упаковок на транспортных сетях, а также при проектировании и анализе передачи сообщений в виртуальных контейнерах в перспективных магистральных опорных сетях передачи данных типа Backbone.

ВВЕДЕНИЕ

Задачи оптимизации упаковок находят широкое применение при проектировании и анализе функционирования многопродуктовых сетей с мелкопартионными дискретными потоками, передаваемыми по сети в некоторых транспортных блоках заданного объема [1]. При этом объемы отдельно взятых потоков, выраженные количеством единиц потока из узлов источников в узлы назначения, значительно меньше объема транспортного блока. Задача оптимизации упаковок заключается в слиянии (объединении) нескольких исходящих из каждого узла потоков с разными адресами назначения в общие транспортные блоки. В качестве критерия оптимизации в таких задачах может быть принят, например, минимум приведенных затрат на передачу и обработку всех потоков, заданных на сети, а в ограничениях может учитываться время на доставку потоков адресату, пропускные способности узлов и ребер сети [2]. При решении задачи некоторые потоки могут несколько раз сливаться в разных узлах с другими потоками, поэтому необходимо иметь такую структуру данных, которая позволяла бы эффективно запоминать и определять узлы слияния и слитые потоки для всех корреспондирующихся пар.

Цель работы — выбор структуры данных для представления процесса слияния и разработка алгоритмов нахождения узлов слияния и слитых потоков.

ПОСТАНОВКА ЗАДАЧИ

Рассмотрим содержательную постановку задачи. Пусть задана неориентированная сеть $G(N,P)$ с множеством узлов N, $n = \left| N \right|$ и множеством ребер

P, $p = |P|$, где n и p соответственно число узлов и ребер сети, а $|\cdot|$ — знак мощности множества. На сети задана целочисленная матрица потоков $A = \| a_{ij} \|_{n\times n}$, $a_{ii} = 0$, $i = \overline{1,n}$, которые подлежат единовременной передаче из источников i в стоки j, $i, j = \overline{1,n}$. Потоки должны передаваться по сети в транспортных блоках объема $\omega >> a_{ij}$. При этом предполагается, что каждый поток может быть упакован в транспортный блок только целиком.

При решении оптимизационной задачи над потоками a_{ij} итеративно выполняются следующие преобразования:

$$a'_{ik} \leftarrow a_{ik} + a_{ij},\ a'_{kj} \leftarrow a_{kj} + a_{ij},\ c_{ij} \leftarrow k,\ a_{ij} \leftarrow 0, \qquad (1)$$

где «$\leftarrow$» знак операции присваивания; k — узел, через который выполняется преобразование; c_{ij} — элементы справочной матрицы слияния потоков $C = \| c_{ij} \|_{n\times n}$, которые определяются так:

$$c_{ij} = \begin{cases} k, \text{ если поток } a_{ij} \text{ сливается с потоком } a_{ik}, \\ i, \text{ если поток } a_{ij} \text{ непосредственно направляется в узел } j, \\ 0, \text{ если } i = j. \end{cases} \qquad (2)$$

На различных шагах работы алгоритма оптимизации необходимо (например, для расчета времени доставки потока адресату) определять последовательность $\Omega_{ij} = \{(i,k_1),(k_1,k_2),...,(k_m,j)\}$ с промежуточными узлами $\{k_1,k_2,...,k_m\}$, в которых выполняется дополнительная обработка каждого из потоков a_{ij} и их общее число $\nu_{ij} = |\{k_1,k_2,...,k_m\}|$. Кроме того, при выборе очередного потока a_{ij} для преобразования (1) может потребоваться нахождение множества других потоков $\{a^*_{ij}\}$, которые уже были слиты с потоком a_{ij} на предыдущих итерациях.

ОСНОВНЫЕ РЕЗУЛЬТАТЫ И АЛГОРИТМ РЕШЕНИЯ

Докажем следующие утверждения.

Теорема 1. Для любого $a_{ij} \in \{a_{ij} \,|\, c_{ij} \neq i \wedge c_{ij} \neq 0\}$ построение матрицы C в соответствии с (2) позволяет найти последовательность Ω_{ij} и установить число дополнительных обработок потока a_{ij}, где «$\wedge$» — знак конъюнкции (логического «и»).

Доказательство. Для доказательства теоремы сначала покажем, что определение Ω_{ij} сводится к построению и последующему прохождению в глубину бинарного дерева с корнем в вершине (i,j). Для этого рассмотрим пример. Пусть поток $(1,9)$ был подвергнут преобразованиям так, как показано на рис. 1. Тогда элементы матрицы C, отражающие это преобразование, будут следующими: $c_{19} = 3$, $c_{13} = 2$, $c_{12} = 1$, $c_{23} = 2$, $c_{39} = 8$,

$c_{89}=8$, $c_{38}=5$, $c_{35}=4$, $c_{34}=3$, $c_{45}=4$, $c_{58}=6$, $c_{56}=5$, $c_{68}=6$. Зная эти элементы, легко построить дерево с корнем в (1,9), описывающее преобразование потока (1,9) (рис. 2). При построении дерева сначала строятся левые поддеревья до тех пор, пока не будет найден элемент $c_{ij}=i$, затем происходит возврат на уровень выше и строятся правые поддеревья. Для любого корня правого поддерева процесс построения поддеревьев может быть продолжен, если для этого корня соответствующий $c_{ij}\neq i$.

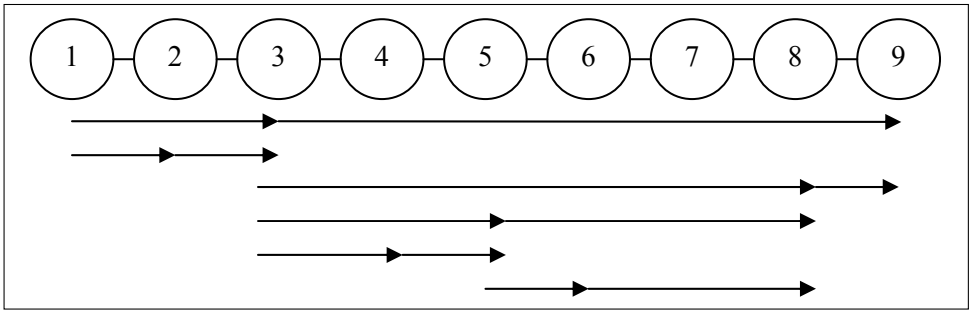

Рис. 1. Пример преобразования потока (1,9)

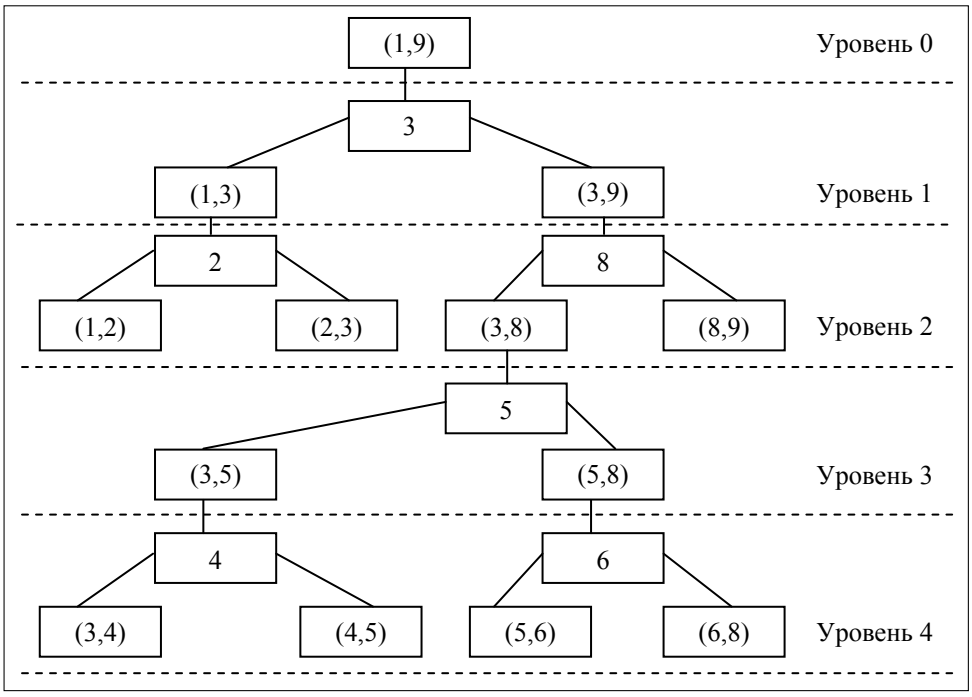

Рис. 2. Дерево преобразования потока (1,9)

Лемма. Листья бинарного дерева, построенного прохождением в глубину с использованием элементов матрицы C, порождают последовательность Ω_{ij} при их просмотре слева направо. Доказательство леммы следует из правил построения бинарного дерева и того факта, что для каждого листа дерева соответствующий $c_{ij}=i$. Последовательность листьев c_{ik_1}, $c_{k_1k_2},\ldots,$ $c_{k_{m-1}k_m}$, c_{k_mj} дает искомую последовательность $\Omega_{ij}=\{(i,k_1),(k_1,k_2),\ldots$

$\ldots,(k_m, j)\}$. Для примера на рис. 1 такая последовательность будет следующей:

$$\Omega_{19} = \{(1,2),(2,3),(3,4),(4,5),(5,6),(6,8),(8,9)\}.$$

Утверждение леммы позволяет построить эффективный алгоритм построения и прохождения бинарного дерева, в котором поддеревья генерируются в порядке их просмотра. Пусть $St(\xi)$ — стек, обслуживаемый по правилу «последний пришел–первый вышел», где ξ — указатель текущего размещения стека; $F(m)$ — вектор, в котором формируется последовательность узлов $\{i, k_1, k_2, ..., k_m, j\}$; «$\vee$» — знак дизъюнкции (логического «или»).

Алгоритм 1.

1. $m \leftarrow 1;\ F(m) \leftarrow i$.
2. $\xi \leftarrow 1;\ St(\xi) \leftarrow j$.
3. $k \leftarrow i;\ l \leftarrow j$.
4. Если $c_{kl} \neq k \wedge ((a_{kl} \neq 0 \wedge \xi = 1) \vee a_{kl} = 0)$, то выполнить 5–7, иначе перейти к 8.
5. $\xi \leftarrow \xi + 1$.
6. $St(\xi) \leftarrow c_{kl}$.
7. $l \leftarrow c_{kl}$; перейти к 4.
8. $m \leftarrow m + 1$.
9. $F(m) \leftarrow l$.
10. Если $\xi > 1$, то перейти к 11, иначе к 14.
11. $k \leftarrow St(\xi)$.
12. $l \leftarrow St(\xi - 1)$.
13. $\xi \leftarrow \xi - 1,$ перейти к 4.
14. $\nu_{ij} \leftarrow m - 2$.
15. Стоп.

В строке 4 алгоритма 1 дополнительные условия $((a_{kl} \neq 0 \wedge \xi = 1) \vee \vee a_{kl} = 0)$ введены для ограничения роста бинарного дерева в случае, если для вновь найденного подкорня (k,l) соответствующий $a_{kl} \neq 0$.

После работы алгоритма значение ν_{ij} равно числу транзитных узлов от i до j, в которых выполняется дополнительная обработка потока a_{ij}.

Вернемся к доказательству теоремы. Для этого рассмотрим связь формирования матрицы C с преобразованием потоковой матрицы A. Преобразование любого потока a_{ij} заключается в выполнении действий (1). Поскольку дробление потока запрещено, потоки $(a_{ik} + a_{ij})$ и $(a_{kj} + a_{ij})$ можно рассматривать как единые потоки, возникшие соответственно в узлах i, k и адресованные в узлы k и j. Применяя операцию (1) для потоков a'_{ik} и a'_{kj} получим новое разбиение, для которого также можно выполнить преобразование. По индукции легко показать, что во всех последующих разбиениях учитывается первоначальный поток a_{ij}. Для любого потока, подвергнутого

преобразованию, значение $c_{ij} = k \neq i$ и указывает узел, в котором поток a_{ij} первый раз был соединен с потоками a_{ik} и a_{kj}. Из леммы следует, что для получения первого узла, в котором выполняется дополнительная обработка потока a_{ij}, необходимо полностью построить каждое левое поддерево, используя в качестве корня дерева элемент c_{ij}. Восстановление всех остальных промежуточных узлов, в которых обрабатывается поток a_{ij}, осуществляется алгоритмом 1 при прохождении всего дерева, что и доказывает теорему.

При работе со справочной матрицей слияния потоков в основном алгоритме оптимизации при выборе очередного потока a_{ij} для преобразования (1) может потребоваться нахождение множества других потоков $\{a_{ij}^*\}$, которые уже были слиты с потоком a_{ij} на предыдущих итерациях, так как преобразование a_{ij} приводит к увеличению времени доставки потока адресату не только для a_{ij}, но и для множества $\{a_{ij}^*\}$. Оказывается, что для определения множества $\{a_{ij}^*\}$ для любого a_{ij}, достаточно воспользоваться справочной матрицей C, формируемой согласно (2).

Теорема 2. Для любого потока a_{ij}, над которым выполняется преобразование (1), множество $\{a_{ij}^*\}$ может быть определено из справочной матрицы C.

Доказательство. Покажем, что задача определения $\{a_{ij}^*\}$ эквивалентна построению некоторого дерева поиска и прохождению его снизу вверх. Прежде всего, отметим, что поток a_{ij} может быть представлен в дереве поиска самым нижним подкорнем (i, j) с двумя листьями: левым — (i,k) и правым — (k, j), где k — узел, через который выполняется преобразование потока a_{ij}. Рассмотрим далее, какие потоки образуют множество $\{a_{ij}^*\}$. Из построения C ясно, что для первых потоков, охватывающих a_{ij}, будут выполняться условия:

$$c_{\xi j} = i, \quad \xi = \overline{1,n}, \tag{3}$$

$$c_{i\xi} = j, \quad \xi = \overline{1,n}, \tag{4}$$

причем (3) означает охват слева, а (4) — охват справа. Для дерева поиска выполнение (3) означает, что для a_{ij} найден левый подкорень, находящийся на уровень выше, а выполнение (4) — что найден правый верхний подкорень. Поскольку в (3) и (4) $\xi = \overline{1,n}$, то может найтись несколько левых и правых подкорней, находящихся на одном уровне дерева. Это говорит о том, что дерево, которое строится, не является бинарным. Для каждого вновь полученного подкорня процесс построения новых подкорней продолжается до тех пор, пока будут выполняться условия (3) или (4).

Заметим, что число дополнительных обработок, определяющее время t_{ij} доставки любого потока, представленного подкорнем в дереве поиска, может быть получено с использованием процедуры построения и прохождения бинарного дерева в глубину, приведенной при доказательстве теоремы 1. Таким образом, как только найден очередной подкорень (k,l), для него необходимо построить бинарное дерево, определяющее последовательность Ω_{kl}, исходя из которой для соответствующего потока a_{kl} может быть рассчитано время доставки.

На рис. 3 показан пример дерева поиска для определения потоков, охватывающих поток a_{58}, который преобразовывается через узел 6 (дерево построено на сети, изображенной на рис. 1). Как видно из рис. 3, для подкорня (5,8) найдены подкорни: (3,8), (3,9), (1,9). Соответствующие им последовательности Ω_{38}, Ω_{39} и Ω_{19} строятся прохождением бинарного дерева в глубину:

$$\Omega_{38} = \{(3,4),(4,5),(5,6),(6,8)\},\ \ \Omega_{39} = \{\Omega_{38},(8,9)\},\ \ \Omega_{19} = \{(1,2),(2,3),\Omega_{39}\}.$$

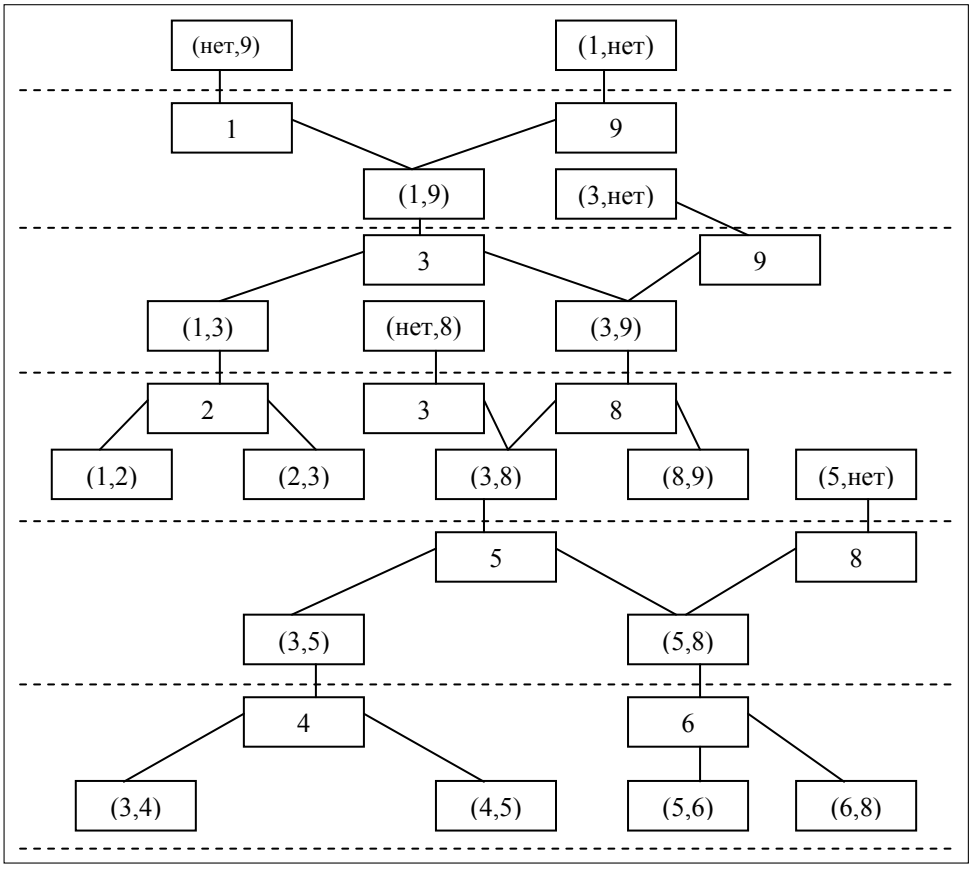

Рис. 3. Дерево поиска потоков, слитых с потоком (5,8)

Приведем рекурсивную процедуру, осуществляющую построение дерева исчерпывающего поиска для определения множества $\{a_{ij}^*\}$.

Алгоритм 2.

1. ВХОД (i, j) рекурсивный.
2. $k \leftarrow i,\ l \leftarrow j$.
3. Для $\{\xi \mid \xi = \overline{1,n}\}$ выполнить 4–14.
4. Если $c_{\xi l} = k \wedge a_{\xi l} = 0,$ то перейти к 5, иначе к 9.
5. $\{a_{ij}^*\} = \{a_{ij}^*\} \cup a_{\xi l}$. Используя алгоритм 1 определить $\Omega_{\xi l}$ и $\nu_{\xi l}$.
6. Рассчитать $t_{\xi l}$.
7. Если $(t_{\xi l} > T_{\xi l} \vee \nu_{\xi l} > \nu_g)$, то $\eta \leftarrow \eta + 1$.
8. Вызвать ВХОД (ξ, l).
9. Если $c_{k\xi} = l \wedge a_{k\xi} = 0,$ то перейти к 10, иначе к 14.
10. $\{a_{ij}^*\} = \{a_{ij}^*\} \cup a_{k\xi}$. Используя алгоритм 1 определить $\Omega_{k\xi}$ и $\nu_{k\xi}$.
11. Рассчитать $t_{k\xi}$.
12. Если $(t_{k\xi} > T_{k\xi} \vee \nu_{k\xi} > \nu_g),$ то $\eta \leftarrow \eta + 1$.
13. Вызвать ВХОД (k, ξ).
14. Перейти к 3.
15. ВЫХОД.

В строках 4 и 9 введены дополнительные условия $a_{\xi l} = 0$ и $a_{k\xi} = 0$ для случая, когда в матрице C имеются элементы $c_{ij} = k,\ k \neq j,$ а соответствующие им потоки $a_{ij} \neq 0,$ что может привести к зацикливанию алгоритма.

В приведенном алгоритме: k, l, ξ — рекурсивные переменные, для которых организовываются поколения данных при каждом новом вхождении в процедуру; «$\cup$» — знак объединения множеств; t_{ij} — расчетное время доставки потоков адресату; ν_{ij} — определяет число промежуточных узлов, в которых выполняется дополнительная обработка соответствующего потока; ν_g — допустимое число дополнительных обработок; η — признак нарушения ограничений на заданное время доставки T_{ij} или число дополнительных обработок потока ν_g. Значение η указывает число таких нарушений.

Анализ алгоритма 2 делает прозрачным доказательство теоремы (которое легко провести по индукции), поскольку на всех уровнях рекурсии и для каждого вновь найденного подкорня выполняется исчерпывающий поиск новых подкорней, для которых справедливо (3) или (4). Это означает, что для любого потока a_{ij} алгоритм 2 корректно строит множество потоков $\{a_{ij}^*\}$, которые были слиты с потоком a_{ij} на предыдущих итерациях основного алгоритма оптимизации упаковок в многопродуктовой сети [2, 3].

ВЫВОДЫ

На основании доказанных утверждений предложены способ построения справочной матрицы слияния потоков и эффективные алгоритмы, позволяющие определять узлы слияния и слитые потоки для всех корреспондирующихся пар в многопродуктовой сети с дискретными мелкопартионными потоками.

Разработанные алгоритмы могут быть использованы при решении различных задач оптимизации упаковок на транспортных сетях с мелкопартионными потоками грузов, а также при проектировании и анализе передачи пакетов сообщений в виртуальных контейнерах в перспективных крупномасштабных общенациональных и интернациональных сетях на основе сверхширокополосных каналов связи и схем типа опорной сети (например, в таких, как Европейская опорная сеть E – bone).

ЛИТЕРАТУРА

1. *Васянин В.А., Трофимчук А.Н.* Задача выбора иерархической структуры многопродуктовой коммуникационной сети с мелкопартионными дискретными потоками // Екологічна безпека та природокористування: Зб. наук. праць. — Київ, 2012. — Вип. 10. — С. 182–204.
2. *Васянин В.А*. Об одной задаче дискретной оптимизации в процессах управления перевозками на мультипотоковых транспортных сетях // Кибернетика и вычислит. техника. — 1983. — Вып. 60. — С. 82–87.
3. *Васянин В.А*. Сравнительная эффективность алгоритмов оптимизации упаковок в мультипотоковых сетях // Дискретные системы управления: Сб. науч. тр. — Киев: Ин-т кибернетики им В.М. Глушкова АН УССР, 1988. — С. 36–45.

Поступила 19.07.2013