ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ

Сборник трудов
1964 г. Института математики СО АН СССР Выпуск II

НЕКОТОРЫЕ АЛГОРИТМЫ ДЛЯ АНАЛИЗА ОРИЕНТИРОВАННЫХ ГРАФОВ

Л.Я. Лейфман, Л.Т. Петрова

В предлагаемой статье рассмотрены четыре алгоритма для анализа и преобразования ориентированных графов. Эти алгоритмы могут быть полезными в задачах анализа сетевых графиков, которые по определению являются ориентированными графами без контуров.

Прежде всего при составлении больших сетевых графиков группой людей возможно появление контуров вследствие ошибок или несогласованности между составителями графика. Поэтому одной из задач анализа сетевого графика является выявление контуров. Алгоритмы для решения этой задачи рассмотрены в настоящей статье.

Другой рассматриваемый в этой работе алгоритм предназначен для упорядочения дуг ориентированного графа по классам. Такое упорядочение полезно, когда необходимо ввести в имеющийся граф новые дуги и вершины или вычертить части графа.

Наконец, при обработке графов большого размера в машинах с недостаточно большим объемом памяти может существенно помочь сокращение размеров графа за счет объединения длинных путей без разветвлений в одну дугу. Один из алгоритмов для такого объединения также рассмотрен в настоящей статье.

§ I. Выявление контуров в ориентированном графе

Пусть имеется ориентированный граф с $\bar{M}$ вершинами и $\bar{N}$ дугами, заданный неупорядоченным списком дуг (i, j). Требуется: I) выявить наличие контуров в данном графе и 2) выписать пути, образующие контуры.

Для решения первой задачи может быть использован следующий алгоритм (A).

На вершинах $m = 1, 2, \ldots, \bar{M}$ данного графа определяется функция

$$\varphi(m) = \begin{cases} \text{I, если вершина } m \text{ принадлежит контуру} \\ \quad \text{или пути, начинающемуся на контуре;} \\ \text{0 - в противном случае.} \end{cases}$$

Вычисление функции φ ведется рекуррентным образом с помощью монотонно убывающей последовательности вспомогательных функций $\psi_k(m) \le \psi_{k-1}(m)$, сходящейся к $\varphi(m)$, а именно, при некотором конечном k_0 будет: $\psi_{k_0-1}(m) \equiv \psi_{k_0}(m) \equiv \varphi(m)$, $m = 1, 2, \ldots, \bar{M}$.

Опишем построение функций $\psi_k(m)$. Положим $\psi_0(m) \equiv 1$. Пусть вычислена $\psi_{k-1}(m)$, $m = 1, 2, \ldots, \bar{M}$. Вычисление $\psi_k(m)$ ведется следующим образом: последовательно просматриваем список дуг (i, j). Если $\psi_{k-1}(i) = 0$, то переходим к следующей дуге. Если же $\psi_{k-1}(i) = 1$, то определяем соответствующее значение $\psi_k(j) = 1$. Всем $\psi_k(m)$, не определенным после полного просмотра списка дуг, приписываем значение 0.

Поясним геометрический смысл рекуррентного построения функции $\varphi(m)$. На первом шаге значение $\psi_1(m) = 1$ получает каждая вершина, в которую входит хотя бы одна дуга (i, m). Следовательно, значения $\psi_1(m) = 0$ будут приниматься на тех вершинах, в которые не входит ни одна дуга. Эти вершины не могут лежать на контуре или на пути, выходящем из контура, поэтому на всех дальнейших шагах для таких вершин сохраняется значение $\psi_k(m) = 0 = \varphi(m)$, и эти вершины в дальнейшем исключаются из рассмотрения вместе с выходящими из них дугами. Таким образом, граф сокращается за счет исключения дуг, выходящих из начальных вершин (т.е. вершин, не имеющих входящих в них дуг). Тогда в графе появляются новые начальные вершины, и процесс продолжается до тех пор, пока граф не останется без начальных вершин. А это будет означать, что из графа исключены все незамкнутые пути, выходящие из начальных вершин, т.е. мы подойдем к контуру, если он имеется в графе.

Если окажется, что $\varphi(m) \equiv 0$, то контуров в графе нет. Если же $\varphi(m) \not\equiv 0$, то граф содержит контуры. Так решается первая задача о выявлении контуров в графе.

В случае, когда граф содержит контуры, строится функция:

$$\varphi^*(m) = \begin{cases} \text{I, если вершина } m \text{ принадлежит контуру или} \\ \quad \text{пути, ведущему в контур;} \\ \text{0 - в противном случае.} \end{cases}$$

Для ее вычисления последовательность $\{\psi_k^*\}$ строится по дугам с обратной ориентацией точно так же, как последовательность $\{\psi_k\}$.

Вершины, в которых $\varphi(m) = \varphi^*(m) = 1$, принадлежат либо одному из контуров, либо путям, ведущим из контура в контур. Обозначим множество таких вершин через M^0.

Ясно, что вычисление функций φ и φ^* можно вести одновременно, строя по каждой дуге (i, j) параллельно $\psi_k(j)$ и $\psi_k^*(i)$. При этом число просмотров списка дуг для окончательного вычисления обеих функций $\varphi(m)$ и $\varphi^*(m)$ не превышает максимального числа L звеньев на незамкнутых путях. Специфика задач анализа сетевых графиков состоит в том, что даже при очень больших сетевых графиках L сравнительно невелико. Это обстоятельство обусловливает достаточную эффективность алгоритма.

Пример приведен на рис. I и табл. I-2. На рис. I изображен исходный граф. Список его дуг приведен в табл. I. В списке вершин в конце табл. 2 знаком ^{0} отмечены вершины, в которых $\varphi(m) = \varphi^*(m) = 1$, т.е. вершины, принадлежащие M^0.

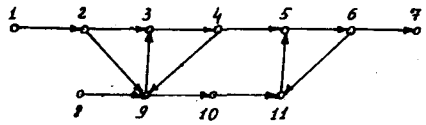
Теперь переходим непосредственно к решению второй задачи - выделению отдельных путей, образующих контур.

Оставим в списке дуг лишь те дуги (i, j), оба конца которых принадлежат M^0, т.е. $\varphi(i) = \varphi^*(i) = 1$ и $\varphi(j) = \varphi^*(j) = 1$. Это множество дуг обозначим N^0. Как ясно из определений функций φ и φ^*, в N^0 входят все те и только те дуги, которые лежат на путях, либо образующих контуры, либо соединяющих их между собой. Таким образом, для каждой вершины $m_0 \in M^0$ найдется в N^0 дуга (m_0, m_1), выходящая из m_0. Действительно, по определению множества вершин M^0 получим $\varphi(m_0) = \varphi^*(m_0) = 1$. Поэтому в N имеются дуги (m_0, m'), выходящие из m_0. Покажем, что хотя бы одна из них принадлежит N^0. Из построения функции $\varphi(m)$ следует неравенство $0 \le \varphi(m_0) \le \varphi(m') \le 1$ для

Таблица I

Список дуг (множество N)

Начало дуги	Конец дуги
I	2
2	3
3	4
4	5
5	6
6	7
2	9
4	9
6	II
8	9
9	I0
I0	II
9	3
II	5



Р и с. I

Таблица 2

m	ψ_0	ψ_1	ψ_2	$\psi_3=\psi_2=\varphi$	ψ_0^*	ψ_1^*	$\psi_2^*=\psi_3^*=\varphi^*$	m
I	I	0	0	0	I	I	I	I
2	I	I	0	0	I	I	I	2
3	I	I	I	I	I	I	I	3^0
4	I	I	I	I	I	I	I	4^0
5	I	I	I	I	I	I	I	5^0
6	I	I	I	I	I	I	I	6^0
7	I	I	I	I	I	0	0	7
8	I	0	0	0	I	I	I	8
9	I	I	I	I	I	I	I	9^0
I0	I	I	I	I	I	I	I	$I0^0$
II	I	I	I	I	I	I	I	II^0

всех дуг (m_0, m'). Следовательно, $\varphi(m') = \varphi(m_0) = 1$ для всех таких m'. С другой стороны, должно быть $\varphi^*(m') = 1$ хотя бы для одного m'. Допустим противное, т.е. что $\varphi^*(m')=0$ для всех m'. Но тогда по построению функции $\varphi^*(m)$ должно быть $\varphi^*(m_0)=0$, что противоречит условию. Таким образом, имеется хотя бы одна дуга $(m_0, m_1) \in N^0$.

Процесс выделения из N^0 пути, образующего контур, должен состоять в следующем.

Выберем из множества N^0 любую дугу (m_0, m_1). Так как $m_1 \in M^0$, то по доказанному в N^0 имеется дуга (m_1, m_2), выходящая из вершины m_1. Аналогично существует дуга $(m_2, m_3) \in N^0$ и т.д. Так как число вершин в M^0 конечно и из каждой из них можно выйти по некоторой дуге из N^0, то после конечного числа q шагов мы придем к некоторой вершине $m_r \in M^0$, которая однажды была уже пройдена на нашем пути. Тем самым мы получим контур, состоящий из дуг

$$(m_r, m_{r+1}), (m_{r+1}, m_{r+2}), \ldots, (m_{q-1}, m_r). \qquad (I)$$

Опишем теперь алгоритм (B), осуществляющий этот процесс достаточно экономным образом.

Просматривая последовательно список дуг N^0, строим следующее отображение ν множества M^0 в себя:

$$\nu(m) = m',$$

где (m, m') – первая из встречающихся в списке N^0 дуг с началом в m. Так как по доказанному выше для каждого $m \in M^0$ есть хотя бы одна такая дуга, то отображение $\nu(m)$ определено на всем множестве M^0. Ясно, что построение отображения ν осуществляется за один просмотр списка N^0.

Теперь, взяв произвольную вершину $m_0 \in M^0$, выписываем дугу (m_0, m_1), где $m_1 = \nu(m_0)$. Затем выписываем дугу (m_1, m_2), где $m_2 = \nu(m_1)$, и т.д. Как показано выше, этот путь содержит контур (I).

Записав найденный контур (I), исключим из N^0 последнюю выписанную дугу (m_{q-1}, m_r) этого контура, разорвав его таким образом.

Теперь применяем к оставшемуся множеству дуг алгоритм (A). Если в этом множестве алгоритмом (A) будут обнаружены контуры, то применяем к нему алгоритм (B) и выделяем еще один контур. Так как при последовательном применении алгоритмов (A) и (B) список дуг сокращается по меньшей мере на одну дугу,

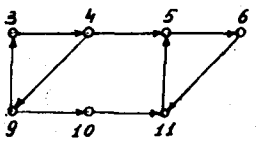
то поочередно применяя алгоритмы (A) и (B), мы после конечного числа шагов исчерпаем весь список дуг N .

Рассмотрим приведенный ранее пример. Список дуг N^0 выписан в таблице 3. Оставшийся граф изображен на рис. 2.

Таблица 3

Множество N^0

Начало дуги	Конец дуги
3	4
4	5
5	6
4	9
6	II
9	I0
I0	II
9	3
II	5



Р и с. 2

Выписываем из табл. 2 множество M^0 в таблицу 4 и записываем в ней отображение $\nu(m)$:

Таблица 4

M^0	$\nu(m)$
3	4
4	5
5	6
6	II
9	I0
I0	II
II	5

Таблица 5

Множество N_1^0

Начало дуги	Конец дуги
3	4
4	5
5	6
4	9
6	II
9	I0
I0	II
9	3

Начнем с первой в списке M^0 вершины 3. Тогда согласно указаниям алгоритма (B) получим путь (3,4), (4,5), (5,6), (6,II), (II,5), содержащий контур (5,6),(6,II), (II,5).

Исключаем из N^0 последнюю дугу (II,5), разрывая контур.

Для следующего применения алгоритма (A) исходное множество дуг, которое мы обозначим N_1^0, приведено в таблице 5.

После применения алгоритма (A) множество оставшихся вершин M_1^0 будет 3,4,9. Повторное применение алгоритма (B) дает еще один контур: (3,4), (4,9), (9,3).

Разорвав этот контур исключением дуги (9,3) и применив еще раз алгоритм (A), мы полностью исчерпаем список дуг N . На этом процесс заканчивается. В результате выписаны все контуры, имевшиеся в графе. Так будет в тех случаях, когда контуры изолированы друг от друга, т.е. каждая дуга может принадлежать не более чем одному контуру.

Если же в графе имеются смежные контуры, т.е. контуры с общими дугами (рис. 3), то по описанному выше процессу могут быть в некоторых случаях выписаны не все контуры, а лишь по одному из группы смежных. Поэтому при анализе сетевого графика после исправления обнаруженных ошибок описанный процесс выявления и выписывания контуров повторяется еще раз.

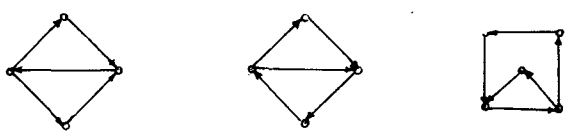


Рис. 3.

Можно ожидать, что ошибки при составлении сетевых графиков, как правило, не будут приводить к появлению больших групп смежных контуров и для выявления всех контуров не потребуется большого числа повторений описанного процесса.

При реализации описанной схемы на ЭВМ размещение материала осуществляется аналогично тому, как при нахождении критического пути в [I].

Значения функций $\psi_{k-1}(m)$ и $\psi_k(m)$; $\psi_{k-1}^*(m)$ и $\psi_k^*(m)$ записываются на месте $T'(m)$ – по одному двоичному разряду на каждую функцию. Здесь же по одному двоичному разряду используется для отметки исключенных из списка дуг. В начале каждого шага при построении функций ψ_k и ψ_k^* на месте всех

их значений записываются нули, заменяемые единицами согласно указаниям алгоритма (A).

Параллельно с $\varphi(m)$ и $\varphi^*(m)$ в случае $m \in M^0$ в тех же ячейках на месте значений времени (не используемых при выявлении контуров) записываются значения $\nu(m)$.

Запись контура, естественно, выдается в виде списка последовательных вершин на нем:

$$m_r, m_{r+1}, \ldots, m_{q-1}, m_r .$$

§ 2. Упорядочение дуг по классам

Вершина m ориентированного графа без контуров относится к классу M_ℓ, если максимальная из длин путей, ведущих в эту вершину, равна ℓ. Дуга (m', m'') этого графа относится к классу N_ℓ, если $m' \in M_\ell$.

В этом и следующем параграфах рассматриваются только ориентированные графы без контуров, что в дальнейшем особо не оговаривается.

Разбиение вершин данного графа на классы $M_\ell (\ell = 0, 1, \ldots, L)$ можно осуществить с помощью алгоритма (A), так как каждая вершина m относится к классу M_ℓ, где

$$\ell = \sum_{k=1}^{L} \psi_k(m). \qquad (2)$$

Это разбиение можно также получить с помощью алгоритмов для нахождения критического пути в сетевом графике, приведенных в [1] и [2].

Если разбиение вершин на классы M_ℓ уже сделано, то все дуги графа могут быть упорядочены по классам (т.е. выписаны в порядке возрастания ℓ) за два просмотра списка дуг. При первом просмотре подсчитывается число дуг в каждом классе N_ℓ, а при втором просмотре производится их упорядочение по классам.

Этот прием упорядочения применен в [1] при упорядочении таблицы результатов по величине резерва времени.

Он может быть полезен также и в других случаях, где требуется сгруппировать большой массив данных в сравнительно небольшое количество групп.

Опишем алгоритм (C) упорядочения дуг по классам применительно к его реализации на трехадресной ЭВМ.

Предварительной частью алгоритма является разбиение вершин $m \in M$ на классы.

Для каждой вершины $m \in M$ через $\ell(m)$ обозначим номер класса M_ℓ, к которому принадлежит m. Вычисление $\ell(m)$ ведется в процессе работы алгоритма (A) следующим образом:

Перед началом работы алгоритма (A) вместо времен $T''(m)$ (см. [1]) выписываем нули, а в процессе работы алгоритма накапливаем там сумму $\sum_{i=1}^{k} \psi_i(m)$. В результате после окончания работы алгоритма (A) на этих разрядах будет получена величина (см. [2])

$$\ell(m) = \sum_{k=1}^{L} \psi_k(m).$$

Теперь переходим непосредственно к алгоритму (C).

I п р о с м о т р. Просматривая список дуг (m, m') по начальным вершинам m и обращаясь к величинам $\ell(m)$, подсчитываем числа $n(\ell)$ $(\ell = 0, 1, \ldots, L-1)$ дуг, содержащихся в классах N_ℓ. Эти числа накапливаем в L ячейках или в меньшем числе ячеек, размещая в каждой из них по нескольку чисел. При относительно небольшом числе классов этот материал не занимает много места в памяти машины.

Найденные числа $n(\ell)$ используем для отметки групп ячеек, предназначенных для хранения списка дуг классов N_ℓ, а именно: если дуги нулевого класса размещаются, начиная с ячейки P_0, то дуги класса N_1 размещаются, начиная с ячейки $P_1 = P_0 + n(0)$ и вообще дуги класса N_ℓ размещаются, начиная с ячейки

$$P_\ell = P_{\ell-1} + n(\ell-1) = P_0 + \sum_{i=0}^{\ell-1} n(i).$$

2-й п р о с м о т р. Каждую дугу (m, m') из списка дуг N переносим в первую свободную ячейку группы, начинающейся с $P_{\ell(m)}$. Таким путем за один просмотр списка дуг все множество N будет упорядочено (сгруппировано) по классам N_ℓ в порядке нумерации классов.

Если оба списка дуг – неупорядоченный и упорядоченный – одновременно в оперативной памяти машины не умещаются, то вместо одного 2-го просмотра потребуется несколько просмотров списка дуг. А именно, если исходный (неупорядоченный) список дуг размещен во внешней памяти машины, а упорядоченный список создается в оперативной памяти по частям и переносится затем во внешнюю память, то число необходимых просмотров определяется следующим образом.

Пусть рабочая часть оперативной памяти машины, отведенная

под списки дуг, составляет P ячеек. Если мы отведем P' ячеек для вызываемых из внешней памяти частей неупорядоченного списка дуг, а $P''=P-P'$ ячеек – для создаваемых частей упорядоченного списка, то для упорядочения всего множества N потребуется число просмотров, равное

$$\left\lceil \frac{N}{P''} \right\rceil + 1 .$$

При этом число обращений к внешней памяти составит для выписывания исходного списка дуг

$$\left(\left\lceil \frac{N}{P''} \right\rceil + 1\right)\left(\left\lceil \frac{N}{P'} \right\rceil + 1\right) \quad \text{и} \quad \left\lceil \frac{N}{P''} \right\rceil + 1$$

для занесения упорядоченного списка, т.е. всего

$$\left(\left\lceil \frac{N}{P''} \right\rceil + 1\right)\left(\left\lceil \frac{N}{P'} \right\rceil + 2\right)$$

обращений к внешней памяти.

При этом на первом просмотре создается часть упорядоченного списка, содержащая классы $N_0, N_1, \ldots, N_s$, такие, что

$$\sum_{\ell=0}^{s} n(\ell) \le P'' < \sum_{\ell=0}^{s+1} n(\ell), \qquad (3)$$

и часть класса N_{s+1}, если левое неравенство в (3) строгое. На следующих просмотрах последовательно создаются дальнейшие части упорядоченного списка.

§ 3. Объединение путей без разветвлений

Путь $W(m_0, m_1, \ldots, m_q)$ назовем путем без разветвлений, если в вершинах $m_1, \ldots, m_{q-1}$ начинается и оканчивается в точности по одной дуге. К начальной и конечной вершинам m_0 и m_q этого пути такое требование не предъявляется.

Путь $W(m_0, m_1, \ldots, m_q)$ без разветвлений будем называть максимальным путем без разветвлений, если в графе не существует пути W' без разветвлений, содержащего W как правильную часть, т.е. $W' \supset W$, $W' \neq W$.

В ряде вопросов представляется полезным преобразование графа с помощью замены каждого максимального пути без разветвлений $W(m_0, m_1, \ldots, m_q)$ одной дугой (m_0, m_q). Однако при такой замене могут возникнуть неразличимые дуги. Так, например, в графе, изображенном на рис. 4, замена пути без разветвлений W (1,2,3,4) одной дугой (1,4) приводит к возникновению двух неразличимых дуг (1,4). Чтобы избежать этого, не осложняя алгоритм, мы будем заменять каждый максимальный путь без разветвлений путем, состоящим из двух дуг: $(m_0, m_1)(m_1, m_q)$

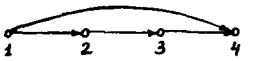


Рис. 4.

Рассмотрим алгоритм ($\mathcal{D}$) для такой замены.

Строим для каждой вершины $m \in M$ два отображения $\nu(m)$ и $\nu^*(m)$ и две функции $\eta(m)$ и $\eta^*(m)$. Построение отображения $\nu(m)$ описано в § 2, отображение ν^* строим точно так же, как ν, поменяв предварительно ориентацию всех дуг графа на обратную.

$$\eta(m) = \begin{cases} 0, \text{ если из } m \text{ выходит не более одной дуги,} \\ 1 \text{ – в противном случае;} \end{cases}$$

$$\eta^*(m) = \begin{cases} 0, \text{ если в } m \text{ входит не более одной дуги,} \\ 1 \text{ – в противном случае.} \end{cases}$$

Отображение $\nu(m)$ определено для всех вершин графа, кроме конечных, $\nu^*(m)$ – для всех вершин графа, кроме начальных.

Построение функций $\eta(m)$ и $\eta^*(m)$ ведется одновременно с построением отображений $\nu(m)$ и $\nu^*(m)$ таким образом: на каждую из них отводим по одному двоичному разряду, в котором первоначально ставим 0. В ходе просмотра списка дуг N, если вершина m встречается в качестве начала дуги, а отображение $\nu(m)$ уже определено (т.е. m встретилась в качестве начала дуги не в первый раз), то полагаем $\eta(m)=1$. Аналогично, просматривая концы дуг при построении ν^*, строим η^*. Таким образом, все построение отображений ν и ν^* и функций η и η^* будет осуществлено за один просмотр списка дуг N.

Далее, за один просмотр списка вершин M будут выделены и заменены все максимальные пути без разветвлений. Просмотр ведется так: вычисляя $H(m)=\eta(m)+\eta^*(m)$, выбираем первую из вершин m', в которой $H(m')=0$. От m' идем к вершине $m''=\nu^*(m')$. Вычисляем $H(m'')$ и, если $H(m'')=0$, то идем далее к вершине $\nu^*(m'')$, и т.д. Этот процесс обрывается на вершине m_0, если $\nu^*(m_0)$ не определена или $H(m_0)\neq 0$. Вершина m_0 является началом максимального пути без разветвлений, на котором лежит вершина m'.

Так как m' не обязательно является концом этого пути, то таким способом искомый путь, вообще говоря, не будет выделен полностью. Поэтому, начиная с m', строим остальную часть этого пути

$$m', \nu(m'), \nu[\nu(m')], \ldots, m_q = \nu[\ldots \nu(m')],$$

идя от вершины к вершине, пока сохраняется $H = 0$ и определено отображение ν.

Найденный максимальный путь без разветвлений

$$W(m_0, m_1, \ldots, m', \ldots, m_q)$$

заменяем двумя дугами: $(m_0, m_1), (m_1, m_q)$.

Выделяя вершины $m \in W$, переопределяем в них η, полагая: $\eta(m) = 1$.

Просматривая таким же образом список вершин дальше, выделяем следующий максимальный путь без разветвлений. Процесс продолжаем до исчерпания множества M.

При реализации алгоритма (D) на ЭВМ, двигаясь от m' к m_0, функцию η не изменяем и запоминаем не весь путь, а лишь последнюю дугу на каждом шаге. Тогда, придя в $m_0 = \nu^*(m_1)$, начинаем движение из m_1, следуя отображению ν, и выписываем искомый путь W:

$$m_0,$$
$$m_1,$$
$$m_2 = \nu(m_1),$$
$$m_3 = \nu(m_2),$$
$$\ldots\ldots\ldots,$$
$$m_q = \nu(m_{q-1}),$$

заменяя на каждом шаге $m_{i+1} = \nu(m_i)$ значения $\eta(m_i)$ на единицу и ставя при m_i отметку F.

Составление нового списка дуг потребует второго просмотра исходного списка. Просматривая начала дуг, обращаемся к списку вершин. Если при вершине m – начале дуги $(m, \tilde{m})$ отметки F нет, то оставляем дугу $(m, \tilde{m})$ в списке и переходим к следующей дуге. Если же отметка F при вершине m имеется, то дуга $(m, \tilde{m})$ исключается из списка. Новые дуги (m_1, m_q) дописываются в конце списка. Дуги (m_0, m_1) при этом не исключаются из списка, так как отметка F при вершинах m_0 отсутствует.

При использовании алгоритма (D) для сокращения сетевого графика временные характеристики для новых дуг в силу их аддитивности получаются по формуле:

$$t_{m_1, m_q} = \sum_{i=1}^{q-1} t_{m_i, m_{i+1}}$$

Литература

[1]. Петрова Л.Т., Карнаухова Н.Н. Об одном алгоритме нахождения критического пути сетевого графика. Стр. (данный сборник).

[2]. Багриновский К.А., Рабинович И.Б. Постановка задачи анализа сетевого графика. Стр. (данный сборник).