В.Н. Бурков, Д.А. Новиков

ЭЛЕМЕНТЫ ТЕОРИИ ГРАФОВ

Теория графов в качестве теоретической дисциплины[1] может рассматриваться как раздел дискретной математики, исследующий свойства конечных множеств (бесконечные графы рассматривать мы не будем) с заданными отношениями между их элементами. Как прикладная дисциплина теория графов позволяет описывать и исследовать многие технические, экономические, биологические и социальные системы.

Задача настоящего материала заключается в том, чтобы, следуя, в основном [8], изложить основные понятия и результаты теории графов, необходимые для постановки и решения задач управления организационными системами.

Изложение материала имеет следующую структуру. В первом разделе вводятся основные понятия, во втором рассматриваются задачи о максимальных путях и контурах на графах, в третьем – свойства псевдопотенциальных графов, в четвертом – задачи о максимальном потоке, в пятом – задачи сетевого планирования и управления.

1. Основные понятия теории графов

Граф – система, которая интуитивно может быть рассмотрена как множество кружков и множество соединяющих их линий (*геометрический способ задания графа* – см. рисунок 1). Кружки называются *вершинами* графа, линии со стрелками – *дугами*, без стрелок – *ребрами*. Граф, в котором направление линий не выделяется (все линии являются ребрами), называется *неориентированным*; граф, в котором направление линий принципиально (линии являются дугами) называется *ориентированным*.

Теория графов может рассматриваться как раздел дискретной математики (точнее – теории множеств), и формальное определе-

[1] *Начало теории графов датируют 1736 г., когда Л. Эйлер решил популярную в то время «задачу о кенигсбергских мостах». Термин «граф» впервые был введен спустя 200 лет* (*в 1936 г.*) *Д. Кенигом.*

ние графа таково: задано конечное множество X, состоящее из n элементов ($X = \{1, 2, ..., n\}$), называемых вершинами графа, и подмножество V декартова произведения $X \times X$, то есть $V \subseteq X^2$, называемое множеством дуг, тогда ориентированным *графом* G называется совокупность (X, V) (неориентированным графом называется совокупность множества X и множества неупорядоченных пар элементов, каждый из которых принадлежит множеству X). Дугу между вершинами i и j, $i, j \in X$, будем обозначать (i, j). Число дуг графа будем обозначать m ($V = (v_1, v_2, ..., v_m)$).

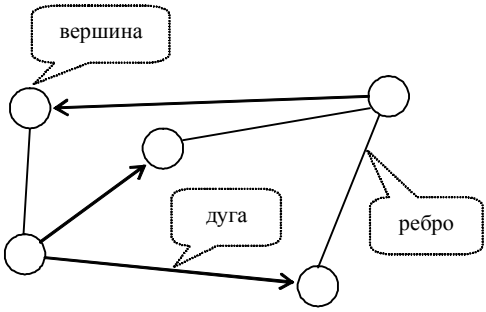

Рис. 1. Пример графа

Язык графов оказывается удобным для описания многих физических, технических, экономических, биологических, социальных и других систем.

Приведем ряд **примеров приложений теории графов**.

1. «*Транспортные» задачи*, в которых вершинами графа являются пункты, а ребрами – дороги (автомобильные, железные и др.) и/или другие транспортные (например, авиационные) маршруты. Другой пример – сети снабжения (энергоснабжения, газоснабжения, снабжения товарами и т.д.), в которых вершинами являются пункты производства и потребления, а ребрами – возможные маршруты перемещения (линии электропередач, газопроводы, дороги и т.д.). Соответствующий класс задач оптимизации потоков грузов, размещения пунктов производства и потребления и т.д., иногда называется *задачами обеспечения* или *задачами о размещении*. Их подклассом являются *задачи о грузоперевозках* [7, 12].

2. *«Технологические задачи»*, в которых вершины отражают производственные элементы (заводы, цеха, станки и т.д.), а дуги –

потоки сырья, материалов и продукции между ними, заключаются в определении оптимальной загрузки производственных элементов и обеспечивающих эту загрузку потоков [7, 12].

3. *Обменные схемы*, являющиеся моделями таких явлений как бартер, взаимозачеты и т.д. Вершины графа при этом описывают участников обменной схемы (цепочки), а дуги – потоки материальных и финансовых ресурсов между ними. Задача заключается в определении цепочки обменов, оптимальной с точки зрения, например, организатора обмена и согласованной с интересами участников цепочки и существующими ограничениями [6, 9, 17].

4. *Управление проектами*[1]. С точки зрения теории графов проект – совокупность операций и зависимостей между ними (*сетевой график* – см. ниже). Хрестоматийным примером является проект строительства некоторого объекта. Совокупность моделей и методов, использующих язык и результаты теории графов и ориентированных на решение задач управления проектами, получила название *календарно-сетевого планирования и управления* (КСПУ) [7, 10]. В рамках КСПУ решаются задачи определения последовательности выполнения операций и распределения ресурсов между ними, оптимальных с точки зрения тех или иных критериев (времени выполнения проекта, затрат, риска и др.).

5. *Модели коллективов и групп*, используемые в социологии, основываются на представлении людей или их групп в виде вершин, а отношений между ними (например, отношений знакомства, доверия, симпатии и т.д.) – в виде ребер или дуг. В рамках подобного описания решаются задачи исследования структуры социальных групп, их сравнения, определения агрегированных показателей, отражающих степень напряженности, согласованности взаимодействия и др.

6. *Модели организационных структур*, в которых вершинами являются элементы организационной системы, а ребрами или дугами – связи (информационные, управляющие, технологические и др.) между ними [13, 18].

---

[1] *Управление проектами – раздел теории управления, изучающий методы и механизмы управления изменениями (проектом называется целенаправленное изменение некоторой системы, осуществляемое в рамках ограничений на время и используемые ресурсы; характерной чертой любого проекта является его уникальность, то есть нерегулярность соответствующих изменений).*

Завершив краткое описание прикладных областей, вернемся к введению **основных понятий** теории графов.

Подграфом называется часть графа, образованная подмножеством вершин вместе со всеми ребрами (дугами), соединяющими вершины из этого множества. Если из графа удалить часть ребер (дуг), то получим *частичный граф*.

Две вершины называются *смежными*, если они соединены ребром (дугой). Смежные вершины называются граничными вершинами соответствующего ребра (дуги), а это ребро (дуга) – *инцидентным* соответствующим вершинам.

Путем называется последовательность дуг (в ориентированном графе), такая, что конец одной дуги является началом другой дуги. *Простой путь* – путь, в котором ни одна дуга не встречается дважды. *Элементарный путь* – путь, в котором ни одна вершина не встречается дважды. *Контур* – путь, у которого конечная вершина совпадает с начальной вершиной. *Длиной пути* (контура) называется число дуг пути (или сумма длин его дуг, если последние заданы).

Граф, для которого из $(i, j) \in V$ следует $(j, i) \in V$ называется *симметрическим*. Если из $(i, j) \in V$ следует, что $(j, i) \overline{\in} V$, то соответствующий граф называется *антисимметрическим.*

Цепью называется множество ребер (в неориентированном графе), которые можно расположить так, что конец (в этом расположении) одного ребра является началом другого. Другое определение: цепь – последовательность смежных вершин. Замкнутая цепь называется *циклом*. По аналогии с простым и элементарным путем, можно определить соответственно *простые и элементарные цепь и цикл*. Любой элементарный цикл является простым, обратное утверждение в общем случае неверно. Элементарная цепь (цикл, путь, контур), проходящая через все вершины графа называется *гамильтоновой цепью* (соответственно – циклом, путем, контуром). Простая цепь (цикл, путь, контур), содержащая все ребра (дуги) графа называется *эйлеровой цепью* (соответственно – циклом, путем, контуром).

Если любые две вершины графа можно соединить цепью, то граф называется *связным*. Если граф не является связным, то его можно разбить на связные подграфы, называемые *компонентами*. *Связностью* графа называется минимальное число ребер, после удаления которых граф становится несвязным. Для ориентирован-

ных графов, если любые две вершины графа можно соединить путем, то граф называется *сильно связным*. Известно [7], что: связность графа не может быть больше, чем $[2m / n]$, где $[x]$ – целая часть числа x; существуют графы с n вершинами и m ребрами, имеющие связность $[2m / n]$; в сильно связном графе через любые две вершины проходит контур.

Связный граф, в котором существует эйлеров цикл, называется *эйлеровым графом*.

В неориентированном графе *степенью вершины i* называется число d_i инцидентных ей ребер. Очевидно, $d_i \le n - 1,\ i \in X$. Граф, степени всех вершин которого равны $n - 1$, называется *полным*. Граф, все степени вершин которого равны, называется *однородным*.

Вершина, для которой не существует инцидентных ей ребер ($d_i = 0$) называется *изолированной*. Вершина, для которой существует только одно инцидентное ей ребро ($d_i = 1$) называется *висячей*.

Известно [7], что: $\sum_{i \in X} d_i = 2\,m$ (данное выражение называется «леммой о рукопожатиях» – поскольку в каждом рукопожатии участвуют две руки, то при любом числе рукопожатий общее число пожатых рук четно (при условии, что каждая рука учитывается столько раз, в скольких рукопожатиях она участвовала)); в любом графе число вершин нечетной степени четно.

Связный граф является эйлеровым тогда и только тогда, когда степени всех его вершин четны (*теорема Эйлера*). Обозначим n_k – число вершин, имеющих степень k, $k = 0, 1, 2, \ldots$. Известно [7, 15], что: $\sum_{k:\, n_k > 0} k\, n_k = 2\,m$.

Для ориентированных графов для каждой вершины можно ввести два числа – *полустепень исхода* d_i^+ (число выходящих из нее вершин) и *полустепень захода* d_i^- (число входящих в нее вершин). В дальнейшем, если не оговорено особо, будем рассматривать графы без *петель*, то есть без дуг, у которых начальная и конечная вершины совпадают. Известно [7, 15], что: $\sum_{i \in X} d_i^- = \sum_{i \in X} d_i^+ = m$; для эйлерова графа имеет место: $d_i^+ = d_i^-$,

$i = \overline{1,n}$; эйлеров граф является объединением контуров, попарно не имеющих общих ребер.

Определим *матрицу смежности* графа как квадратную матрицу $n \times n$, элемент a_{ij} которой равен единице, если $(i, j) \in V$, и нулю, если $(i, j) \overline{\in} V$, $i, j \in X$. Для неориентированного графа матрица смежности всегда симметрическая.

Определим *матрицу инциденций для ребер* графа как прямоугольную матрицу $n \times m$, элемент r_{ij} которой равен единице, если вершина i инцидентна ребру j, и нулю в противном случае, $i = \overline{1,n}$, $j = \overline{1,m}$. Аналогично определяется *матрица инциденций для дуг* графа – как прямоугольная матрицу $m \times n$, элемент r_{ij} которой равен плюс единице, если дуга U_j исходит из вершины i, минус единице, если дуга U_j заходит в вершину i, и нулю в остальных случаях, $i = \overline{1,n}$, $j = \overline{1,m}$

Деревом называется связный граф без простых циклов, имеющий не менее двух вершин. Для дерева $m = n - 1$, а число висячих вершин равно $n_1 = 2 + \sum_{i \geq 2} (i-2)\, n_i$. Легко показать, что в дереве любые две вершины связаны единственной цепью.

Прадеревом называется ориентированное дерево, у которого одна из вершин, называемая *корнем*, не имеет заходящих дуг, а степени захода остальных вершин равны единице.

Плоским (*планарным*) называется граф, который можно изобразить на плоскости так, что различным вершинам соответствуют различные кружки и никакие два ребра не имеют общих точек, отличных от их границ (не пересекаются). Для плоского графа существует понятие *грани* – части плоскости, ограниченной ребрами и не содержащей внутри себя ни вершин, ни ребер. Для простоты определения грани в дальнейшем в основном будем рассматривать графы без висячих вершин. Например, дерево имеет всего одну внешнюю грань – всю плоскость. *Степенью грани* называется число ее граничных ребер (висячие ребра считаются дважды). Обозначим p – число граней плоского графа, p_k – число его граней, имеющих степень k, q_i – степень i-ой грани. Можно показать, что имеет место $\sum_{i=1}^{p} q_i = 2m$, $\sum_{k:\, p_k > 0} k\, p_k = 2m$, $n + p = m + 2$ – *формула*

Эйлера [7, 15]. Данные выражения являются необходимыми условиями существования плоских графов с заданными наборами чисел $\{n_i\}$ и $\{p_i\}$.

Любому связному плоскому графу G можно поставить в соответствие *двойственный* ему связный плоский граф G^*, определяемый следующим образом: каждой грани графа G соответствует вершина графа G^*, каждому ребру V графа G, являющемуся граничным для граней z_1 и z_2, соответствует ребро V^* графа G^*, соединяющее соответствующие граням z_1 и z_2 вершины. Понятие двойственного графа тесно связано с понятием двойственности в линейном программировании [7].

2. Экстремальные пути и контуры на графах

Задачи поиска кратчайших и длиннейших путей на графах возникают в различных областях управления. Сначала мы рассмотрим задачи о кратчайшем пути, затем задачи об экстремальных контурах.

Задача о кратчайшем пути. Пусть задана *сеть* из $n + 1$ вершины, то есть ориентированный граф, в котором выделены две вершины – вход (нулевая вершина) и выход (вершина с номером n). Для каждой дуги заданы числа, называемые длинами дуг. *Длиной пути* (*контура*) называется сумма длин входящих в него дуг (если длины дуг не заданы, то длина пути (контура) определяется как число входящих в него дуг). Задача заключается в поиске кратчайшего пути (пути минимальной длины) от входа до выхода сети[1].

Известно [7, 15], что для существования кратчайшего пути необходимо и достаточно отсутствия в сети контуров отрицательной длины.

Предположим, что в сети нет контуров. Тогда всегда можно пронумеровать вершины таким образом, что для любой дуги (i, j) имеет место $j > i$. Такая нумерация называется *правильной*. Легко показать, что в сети без контуров всегда существует правильная нумерация.

---

[1] *В дальнейшем будем предполагать, что в любую вершину сети можно попасть из входа, и из любой вершины можно попасть в выход* (*вершины, не удовлетворяющие этому требованию, можно удалить*).

Обозначим l_{ij} – длину дуги $(i; j)$. Кратчайший путь в сети, имеющей правильную нумерацию, определяется следующим алгоритмом.

Алгоритм 1.

Шаг 0: Помечаем нулевую вершину индексом $\lambda_0 = 0$;

Шаг k: помечаем вершину k индексом $\lambda_k = \min\limits_{i<k} (\lambda_i + l_{ik})$.

Индекс выхода λ_n будет равен длине кратчайшего пути[1]. На рисунке 2 приведен пример применения алгоритма 1 для определения кратчайшего пути (числа у дуг равны длинам дуг, индексы вершин помещены в квадратные скобки, кратчайший путь выделен двойными линиями).

Когда индексы (называемые в некоторых задачах *потенциалами вершин*) установятся, кратчайший путь определяется методом обратного хода от выхода к входу, то есть кратчайшим является путь $\mu = (0; i_1; i_2; \ldots; i_{n-1}; n)$, такой, что $l_{i_{n-1}n} = \lambda_n - \lambda_{n-1}$ и т.д.

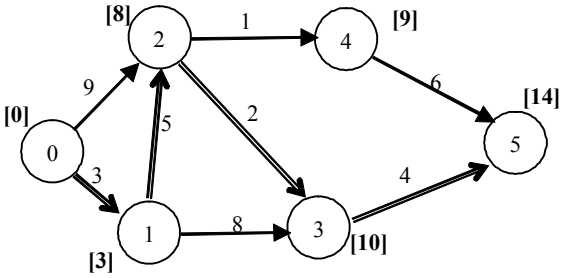

Рис. 2. Поиск кратчайшего пути

Следующий алгоритм дает возможность определять кратчайший путь в общем случае (то есть при произвольной нумерации вершин).

---

[1] *Алгоритм 1 для задач динамического программирования отражает принцип оптимальности Беллмана: если ищется кратчайший путь между двумя точками, то длина пути между любыми двумя точками кратчайшего пути также должна быть минимальна.*

Алгоритм 2 (*алгоритм Форда*).

Шаг 0: Помечаем нулевую вершину индексом $\lambda_0 = 0$, все остальные вершины индексами $\lambda_i^0 = +\infty$, $i = \overline{1,n}$;

Шаг k: Рассматриваем все дуги. Если для дуги $(i; j)$, $\lambda_j^{k-1} - \lambda_i^{k-1} > l_{ij}$, то вычисляем новое значение $\lambda_j^k = \lambda_i^{k-1} + l_{ij}$.

Индексы устанавливаются за конечное число шагов. Обозначим $\{\lambda_i^*\}$ – установившиеся значения индексов, которые обладают следующим свойством: величина λ_i^* равна длине кратчайшего пути из нулевой вершины в вершину i. Кратчайший путь из вершины 0 в вершину i определяется методом обратного хода.

Если длины всех дуг неотрицательны, то для поиска кратчайшего пути применим следующий алгоритм.

Алгоритм 3.

Шаг 0: Помечаем нулевую вершину индексом $\lambda_0 = 0$;

Шаг k: Пусть уже помечено некоторое множество вершин. Обозначим Q – множество непомеченных вершин, смежных с помеченными. Для каждой вершины $k \in Q$ вычисляем величину $\xi_k = \min(\lambda_i + l_{ik})$, где минимум берется по всем помеченным вершинам i, смежным с вершиной k. Помечаем вершину k, для которой величина ξ_k минимальна, индексом $\lambda_k = \xi_k$.

Подобную процедуру повторяем до тех пор, пока не будет помечена вершина n. Длина кратчайшего пути равна λ_n, а сам кратчайший путь определяется так, как это было описано выше.

Запишем задачу о кратчайшем пути как *задачу линейного программирования* (ЛП). Пусть $x_{ij} = 1$, если дуга $(i; j)$ входит в путь[1] μ, $x_{ij} = 0$, если дуга $(i; j)$ не входит в путь μ, $i, j = \overline{0,n}$.

Задачу о минимальном пути можно записать в виде[2]:

$$(1)\ L(x) = \sum_{i,j=0}^{n} l_{ij} x_{ij} \rightarrow \min_x$$

---

[1] *Будем считать, что имеются две дуги между каждой парой вершин, так как, если их нет в исходном графе, то, положив их длину равной бесконечности, мы заведомо исключим их из решения.*

[2] *Ограничение* (*2*) *отражает требование того, что в искомом пути из входа выходит одна дуга и в выход заходит одна дуга. Ограничение* (*3*) *обеспечивает равенство числа заходящих и выходящих в любую промежуточную вершину дуг.*

(2) $\sum_j x_{0j} = 1,\ \sum_j x_{jn} = 1,$

(3) $\sum_i x_{ki} = \sum_j x_{jk}$, $k = \overline{1, n-1}$.

Любое решение системы неравенств (2)-(3) определяет путь в сети без контуров (но не в сети с контурами).

Пусть все контуры имеют строго положительную длину, то есть нет контуров отрицательной и нулевой длины. Тогда решение задачи (1)-(3) определяет путь кратчайшей длины.

Сформулируем задачу ЛП, двойственную задаче (1)-(3), поставив в соответствие ограничениям (2) двойственные переменные λ_0 и λ_n, а ограничениям (3) – двойственные переменные $\{\lambda_i\}$, $i = \overline{1, n-1}$:

(4) $\lambda_n - \lambda_0 \to \max$

(5) $\lambda_j - \lambda_i \le l_{ij}$, $i, j = \overline{0, n}$.

По теореме двойственности линейного программирования [7], для оптимальных решений задач (1)-(3) и (4)-(5) значения целевых функций совпадают.

Задача (4)-(5) называется *задачей о потенциалах* вершин графа. Общая ее формулировка такова: найти потенциалы вершин $\{\lambda_i\}$, удовлетворяющие системе неравенств (5) и максимизирующие некоторую функцию $\Phi(\lambda)$, где $\lambda = (\lambda_0, \lambda_1, ..., \lambda_n)$. Примером является *задача о ближайших потенциалах*, в которой $\Phi(\lambda) = \sum_j |\lambda_j - \lambda_j^0|$, где $\{\lambda_j^0\}$ могут интерпретироваться как желательные потенциалы.

Аналогично задаче о кратчайшем пути формулируется и решается *задача о максимальном* (*длиннейшем*) *пути* – достаточно изменить знаки дуг на противоположные и решить задачу о кратчайшем пути. Для существования решения задачи о максимальном пути необходимо и достаточно отсутствия контуров положительной длины.

В задаче поиска *пути максимальной надежности* длины дуг интерпретируются, например, как вероятности того, что существует связь между соответствующими двумя пунктами. Заменяя длины дуг их логарифмами, взятыми с обратными знаками, получаем,

что путь максимальной надежности в исходном графе будет соответствовать кратчайшему пути в новом графе.

Гораздо более сложными (NP-полными[1]) являются задачи поиска элементарных путей кратчайшей (максимальной) длины в случае, когда в сети имеются контуры отрицательной (соответственно, положительной) длины[2]. Эффективных (не сводящихся к полному перебору) точных алгоритмов для них не существует.

К таким же сложным задачам относятся и задачи поиска кратчайших или длиннейших путей или контуров, проходящих через все вершины графа (элементарный путь (контур), проходящий через все вершины графа, называется *гамильтоновым путем* (*контуром*)). Классическим примером задачи поиска гамильтонова контура является *задача коммивояжера*, заключающаяся в следующем. Коммивояжер (бродячий торговец) должен посетить n городов, побывав в каждом ровно один раз, и вернуться в исходный пункт своего путешествия. Заданы неотрицательные длины дуг, интерпретируемые как расстояние между городами или стоимости проезда. Требуется найти гамильтонов контур минимальной длины (в графе из n вершин существует $n!$ гамильтоновых контуров).

Алгоритмы решения задачи о кратчайшем пути позволяют решать широкий класс задач дискретной оптимизации. В качестве примера приведем задачу целочисленного линейного программирования – *задачу о ранце* (*о рюкзаке*), к которой сводятся многие практически важные задачи определения оптимальной комбинации факторов при ограничениях на общий вес, площадь, объем, финансирование и т.д.

Задача о ранце. Пусть имеется n предметов, которые могут быть полезны в походе. Полезность i-го предмета оценивается

---

[1] *Качественно, если n – число вершин графа, то, если сложность* (*количество вычислений, операций, шагов и т.д.*) *алгоритма поиска точного решения пропорциональна* n^{α}, *где* α *– некоторое положительное число, то говорят, что алгоритм имеет полиномиальную сложность. Если сложность пропорциональна* α^n, *то имеет место экспоненциальная сложность* (*NP-полнота*).

[2] *Существуют несколько алгоритмов проверки отсутствия контуров отрицательной* (*или положительной*) *длины: изменять индексы, пока число шагов алгоритма не превысит максимально необходимое* (*равное* $m \cdot n$) *число; ограничить потенциалы вершин заданными числами* d_i *и при* $\lambda_i \le d_i$ ($\lambda_i \ge d_i$) *проверять действительно ли полученное значение потенциала соответствует длине некоторого пути, или имеется контур отрицательной* (*положительной*) *длины; и др.*

числом a_i, вес предмета (или его объем) – b_i. Суммарный вес, который может нести турист (объем рюкзака), ограничен величиной R. Требуется найти набор предметов, обладающий максимальной суммарной полезностью и удовлетворяющий ограничению.

Обозначим x_i – переменную, принимающую значение ноль (если i-ый предмет не кладется в ранец) или единица (если i-ый предмет кладется в ранец). Тогда задача о ранце имеет вид:

$$(6)\ \sum_{i=1}^{n} a_i x_i \ \rightarrow \max_x$$

$$(7)\ \sum_{i=1}^{n} b_i x_i \ \le R.$$

Верхняя оценка числа возможных комбинаций – 2^n. Однако для решения задачи о ранце существует эффективный алгоритм – *метод динамического программирования*. При его использовании строится сеть (см. примеры в [7, 8, 12]) по следующим правилам. По оси абсцисс будем последовательно откладывать номера предметов, по оси ординат – их вес. Из каждой точки (начиная с точки (0; 0)) выходят две дуги – горизонтальная (соответствующая альтернативе «не брать предмет») и наклонная (соответствующая альтернативе «взять предмет»), вертикальная проекция которой равна весу предмета. Длины наклонных дуг положим равными ценности предметов, длины горизонтальных дуг – нулю. Полученная сеть (конечная вершина является фиктивной и вес любой дуги, соединяющей ее с другими вершинами, равен нулю) обладает следующими свойствами: любому решению задачи (6)-(7) соответствует некоторый путь в этой сети; любому пути соответствует некоторое решение задачи. Таким образом, задача свелась к нахождению пути максимальной длины.

Задача поиска контура минимальной длины решается следующим образом. Если известно, что искомый контур содержит некоторую вершину, то нужно определить кратчайшей путь от этой вершины до нее же, применяя описанные выше алгоритмы. Так как в общем случае контур минимальной длины может проходить через любую вершину графа, то находятся контуры минимальной длины, проходящие через каждую вершину, и среди них выбирается кратчайший. Более простым является следующий алгоритм 4: берется первая вершина (в произвольном их упорядочении) графа и рассматривается сеть, в которой эта вершина явля-

ется одновременно конечной и начальной вершиной. Для этой сети (применением описанного выше алгоритма) ищется путь μ_1 минимальной длины $L(\mu_1)$. Затем первая вершина отбрасывается, и минимальный путь μ_2 ищется для сети, в которой начальной и конечной вершиной является вторая вершина. Затем отбрасывается вторая вершина и т.д. для всех вершин исходного графа, для которых существует контур, проходящий через них и через вершины с большими номерами. Контуром минимальной длины будет контур μ_{min}, длина которого равна $L(\mu_{min}) = \min \{L(\mu_1), L(\mu_2), ..., L(\mu_n)\}$.

Задача поиска контура минимальной средней длины заключается в поиске контура, для которого минимально отношение его длины к числу содержащихся в нем дуг. Для решения этой задачи используется алгоритм 5:

1. Определяем произвольный контур. Пусть L – длина этого контура, k – число его дуг. Вычисляем $l_{cp} = L / k$ и добавляем $(-l_{cp})$ к длинам l_{ij} всех дуг.

Затем определяем контур отрицательной длины, повторяем шаг 1, и т.д. до тех пор, пока на очередном шаге таких контуров не найдется.

Так как на каждом шаге длины всех дуг изменялись на одно и то же число, то на последнем шаге длина дуги равна $l_{ij} - \eta$, где η – суммарное изменение длины каждой дуги на всех шагах.

Значение η равно минимальной средней длине дуг контуров графа. При этом контуром минимальной средней длины является контур, определенный на предпоследнем шаге.

Путь максимальной эффективности. Пусть задана сеть, в которой для каждой дуги $(i; j)$ определены два числа $(Э_{ij}; S_{ij})$, интерпретируемые как эффект при осуществлении соответствующей операции – $Э_{ij}$ и затраты на эту операцию – S_{ij}. Эффективность $K(\mu)$ пути μ определяется как отношение его эффекта $Э(\mu) = \sum_{\mu} Э_{ij}$ к затратам $S(\mu) = \sum_{\mu} s_{ij}$, то есть $K(\mu) = Э(\mu) / S(\mu)$. Задача заключается в поиске пути μ^* максимальной эффективности: $K(\mu) \to \max$.

Если решение $K^* = K(\mu^*)$ этой задачи известно, то по определению K^* выполнено:

(8) $Э(\mu) - K^* S(\mu) \le 0 \quad \forall \mu$

Следовательно, задача свелась к поиску минимального значения K^*, для которого имеет место (8). Другими словами, необходимо найти минимальное K^*, такое, что все пути (длина которых определяется как $l_{ij}(K^*) = Э_{ij} - K^* S_{ij}$) в сети имеют неположительную длину (неравенство (8) должно выполняться, в том числе, и для пути максимальной длины).

Алгоритм 6. 1) Положим $K^* = 0$. Находим путь μ_1 максимальной длины. Положим $K_1 = Э(\mu_1) / S(\mu_1)$ (заметим, что при $K = K_1$ длина пути $\mu(K_1)$ равна нулю).

2) Находим максимальный путь μ_2 при $K = K_1$. Если длина пути μ_2, которую мы обозначим $L(K_1)$, равна нулю, то задача решена. Если $L(K_1) > 0$, то вычисляем $K_2 = Э(\mu_2) / S(\mu_2)$? и находим максимальный путь μ_2 при $K = K_2$ и т.д.

Путь максимальной эффективности с учетом штрафов. Пусть для каждой дуги $(n + 1)$-вершинной сети заданы два числа: эффект $Э_{ij}$ и время t_{ij}. Каждый путь μ из начальной вершины в конечную вершину характеризует некоторый процесс (например, проект). Под продолжительностью пути будем понимать сумму времен его дуг. Если продолжительность процесса отличается от заданного времени T, то налагаются штрафы $\chi(\mu)$, пропорциональные отклонению, то есть: $\chi(\mu) = \begin{cases} \alpha(T - T(\mu)), T(\mu) \le T \\ \beta(T(\mu) - T), T \le T(\mu) \end{cases}$, где коэффициенты α и β могут быть как положительными, так и отрицательными.

Задача заключается в том, чтобы найти путь μ^*, максимизирующий разность между эффектом и штрафами, то есть

$$\mu^* = \arg \max_{\mu} [Э(\mu) - \chi(\mu)].$$

Обозначим $l_{ij}(\lambda) = Э_{ij} - \lambda t_{ij}$, где λ – некоторый параметр, $T(\lambda)$ – продолжительность оптимального пути при параметре λ, то есть пути, имеющего максимальную длину, измеряемую в $l_{ij}(\lambda)$. Легко показать, что с ростом λ величина $T(\lambda)$ не возрастает.

Обозначим $T(\alpha)$, $T(\beta)$ – продолжительности оптимального пути при λ равном α и, соответственно, β; $\mu(\alpha)$, $\mu(\beta)$ – эти пути (для их нахождения необходимо решить две задачи на поиск пути максимальной длины). Рассмотрим шесть случаев (исходную

задачу можно разбить на две подзадачи – поиска максимума $Э(\mu)$ – $\chi(\mu)$ при $T(\mu) \le T$ и при $T(\mu) \ge T$).

Пусть $\alpha \ge \beta$, тогда $T(\beta) \ge T(\alpha)$ и:

1) если $T(\beta) \ge T(\alpha) \ge T$, то $\mu(\beta)$ – оптимальное решение;

2) если $T \ge T(\beta) \ge T(\alpha)$, то $\mu(\alpha)$ – оптимальное решение;

3) если $T(\beta) \ge T \ge T(\alpha)$, то, сравнивая $\mu(\alpha)$ и $\mu(\beta)$ по длинам $l = Э - \chi$, выбираем путь, имеющий максимальную длину.

Пусть $\alpha \le \beta$, тогда $T(\beta) \le T(\alpha)$ и:

4) если $T(\alpha) \ge T(\beta) \ge T$, то $\mu(\beta)$ – оптимальное решение;

5) если $T \ge T(\alpha) \ge T(\beta)$, то $\mu(\alpha)$ – оптимальное решение;

6) если $T(\alpha) \ge T \ge T(\beta)$, то задача не имеет эффективных методов решения (возможные подходы описаны в [7]).

3. Псевдопотенциальные графы

Полный, $(n+1)$-вершинный, симметричный граф называется *псевдопотенциальным*, если длина его любого гамильтонова контура равна одному и тому же числу. Обозначим ℓ_{ij}, $i, j = \overline{0,n}$ – длины дуг.

Известно [7, 8], что для того, чтобы граф был псевдопотенциальным, необходимо и достаточно существование чисел α_i, β_i, $i = \overline{0,n}$, таких, что $\ell_{ij} = \beta_j - \alpha_i$ для всех $i, j = \overline{0,n}$; а также, что любой подграф псевдопотенциального графа является псевдопотенциальным.

Будем считать, что $\alpha_0 = 0$. Обозначим $M_j(\mu) = \sum_{k=1}^{j} \left(\beta_{i_k} - \alpha_{i_{k-1}} \right)$ – сумма длин первых j дуг гамильтонова контура μ, $\gamma_j = \alpha_j - \beta_j$. Определим $M(\mu) = \max\limits_{1 \le j \le n} M_j(\mu)$.

Известно [7, 8], что существует оптимальное решение задачи (1) $M(\mu) \to \min\limits_{\mu}$,

в котором сначала идут вершины с $\gamma_i \ge 0$ в порядке возрастания величин β_i, а затем вершины с $\gamma_i \le 0$ в порядке убывания величин α_i.

Для доказательства этого утверждения обозначим $M_{min} = \min\limits_{\mu} M(\mu)$, а $\mu = (0, i_1, i_2, \ldots, i_n, 0)$ – оптимальный гамильтонов контур (решение задачи (1)). Тогда для него имеет место следующая система неравенств:

$$(2)\quad \begin{cases} M_{\min} \geq \beta_{i_1} \\ M_{\min} + \gamma_{i_1} \geq \beta_{i_2} \\ M_{\min} + \gamma_{i_1} + \gamma_{i_2} \geq \beta_{i_3} \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ M_{\min} + \gamma_{i_1} + \ldots + \gamma_{i_{n-1}} \geq \beta_{i_n} \end{cases}.$$

Если для некоторого индекса $s \in \{1, 2, \ldots, n\}$ выполнено $\gamma_{i_{s-1}} \leq 0,\ \gamma_{i_s} \geq 0$, то из (2) следует справедливость соответствующей системы неравенств для гамильтонова контура $(0, i_1, \ldots, i_{s-2}, i_s, i_{s-1}, i_{s+1}, \ldots, i_n)$. Поэтому всегда существует оптимальное решение, в котором сначала обходятся вершины с положительными γ, а затем с отрицательными γ (вершину i с $\gamma_i = 0$ можно отнести в любую группу).

Если $\gamma_{i_{s-1}} \geq 0,\ \gamma_{i_s} \geq 0$ и $\alpha_{i_{s-1}} < \alpha_{i_s}$, то из (2) следует справедливость соответствующей системы неравенств для гамильтонова контура $(0, i_1, \ldots, i_{s-1}, i_s, i_{s+1}, \ldots, i_n)$.

Наконец, если $\gamma_{i_{s-1}} \leq 0,\ \gamma_{i_s} \leq 0$ и $\beta_{i_{s-1}} > \beta_{i_s}$, то из (2) следует справедливость соответствующей системы неравенств для гамильтонова контура $(0, i_1, \ldots, i_{s-1}, i_s, i_{s+1}, \ldots, i_n)$. Докажем, например, последнее утверждение. Из (2) получаем, что

$$M_{min} + \sum_{j=1}^{s-2} \gamma_{i_j} \geq \beta_{i_{s-1}} = \alpha_{i_{s-1}} - \gamma_{i_{s-1}},$$

$$M_{min} + \sum_{j=1}^{s-2} \gamma_{i_j} + \gamma_{i_{s-1}} \geq \beta_{i_s} = \alpha_{i_s} - \gamma_{i_s}.$$

Но тогда тем более имеет место:

$$M_{min} + \sum_{j=1}^{s-2} \gamma_{i_j} + \gamma_{i_{s-1}} \geq \alpha_{i_s}, \text{ так как } \gamma_{i_s} \leq 0,$$

$$M_{min} + \sum_{j=1}^{s-2} \gamma_{i_j} + \gamma_{i_s} + \gamma_{i_{s-1}} \geq \alpha_{i_s} \geq \alpha_{i_{s-1}}.$$

Итак, пусть $\mu = (0, i_1, i_2, \ldots, i_n, 0)$ – контур, являющийся решением задачи (1), то есть

$$\gamma_{i_j} \geq 0,\ j = 1, 2, \ldots, s, \text{ причем } \beta_{i_1} \leq \beta_{i_2} \leq \ldots \leq \beta_{i_s},$$

$$\gamma_{i_j} \leq 0,\ j = s+1, \ldots, n, \text{ причем } \alpha_{i_{s+1}} \geq \alpha_{i_{s+2}} \geq \ldots \geq \alpha_{i_n}.$$

Тогда

$$(3)\ M_{\min} = \max\left[\beta_{i_1}, \max_{1 \leq k < n}\left(\beta_{i_{k+1}} - \sum_{j=1}^{k} \gamma_{i_j}\right)\right].$$

Частным случаем псевдопотенциального графа является *потенциальный граф*, у которого длина любого гамильтонова контура равна нулю[1]. В частности, потенциальный граф обладает следующими свойствами:

- у потенциального графа длина любого контура равна нулю;
- для n-вершинного потенциального графа существуют числа $\{\lambda_i\}$, такие, что $l_{ij} = \lambda_j - \lambda_i$, $i, j = \overline{1,n}$.
- у потенциального графа для любой вершины сумма длин заходящих дуг по абсолютной величине равна сумме длин исходящих дуг.

4. Задачи о максимальном потоке

Рассмотрим сеть из $(n + 1)$ вершины. Пусть каждой дуге поставлено в соответствие число c_{ij}, называемое *пропускной способностью* дуги $(i; j)$.

Потоком x в сети называется совокупность чисел $\{x_{ij}\}$, где x_{ij} – поток по дуге $(i; j)$, удовлетворяющих условиям $0 \leq x_{ij} \leq c_{ij}$,

[1] *Потенциальный граф может рассматриваться как модель электрической сети, а его свойства как теоретико-графовые аналоги законов Кирхгофа.*

$i, j = \overline{0,n}$, $\sum_j x_{ij} = \sum_k x_{ki}$, $i \neq 0, n$. *Величиной потока* x называется $\Phi(x) = \sum_i x_{0i} = \sum_i x_{in}$.

Задача о максимальном потоке заключается в определении потока максимальной величины[1].

Разрезом W в сети называется любое множество вершин, обязательно содержащее выход и не содержащее вход. Пропускной способностью $C(W)$ разреза W называется сумма пропускных способностей дуг, заходящих в разрез.

Известно [5, 7, 19], что величина любого потока не превышает пропускной способности любого разреза (*теорема Форда-Фалкерсона*).

Следовательно, если удастся найти поток, величина которого равна пропускной способности некоторого разреза, то этот поток максимален, а разрез минимален.

Алгоритм 7 (*алгоритм Форда-Фалкерсона*). Применение алгоритма проиллюстрируем примером сети, приведенной на рисунке 3, в которой пропускные способности всех дуг равны единице.

Шаг 0. Берем произвольный поток (например, поток $x_{01} = x_{12} = x_{25} = 1$). Помечаем начальную вершину индексом «0».

Обозначим Z – множество помеченных вершин.

Общий шаг. Первое действие. Помечаем вершину j индексом $+i$, если, во-первых, существует дуга $(i; j)$, и, во-вторых, $i \in Z$, $j \notin Z$, $x_{ij} < C_{ij}$.

Если в результате этого типа пометок мы пометили выход, то поток можно увеличить хотя бы на единицу (если c_{ij} – целые числа). Двигаясь обратно, можно найти путь, поток по которому можно увеличить. Однако, как видно из примера, этого недостаточно для нахождения максимального потока.

[1] *Наиболее распространенной содержательной интерпретацией является перевозка грузов из начальной вершины в конечную по дугам графа, где пропускная способность дуги характеризует максимальное количество груза, которое по ней можно перевозить в единицу времени.*

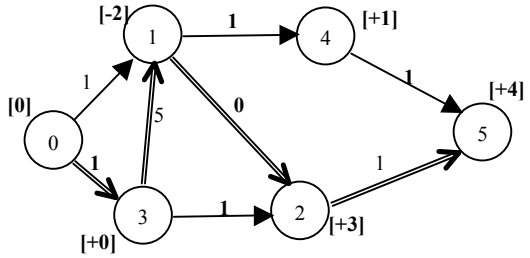

Рис. 3. Поиск максимального потока

Второе действие. Помечаем вершину i индексом $-j$, если, во-первых, существует дуга $(j; i)$, и, во-вторых, $j \in Z,\ i \notin Z,\ x_{ij} > 0$ (легко видеть, что пометки первого типа увеличивают поток по дуге, а пометки второго типа – уменьшают).

Если в результате этого типа пометок мы пометили выход, то поток можно увеличить. Двигаясь обратно, можно найти цепь, в которой каждая вершина помечена номером предыдущей (знак пометки не важен).

Рассмотрим цепь $\mu = (0; 3; 2; 1; 4; 5)$, приведенную на рисунке 3. Полученные в результате второго действия потоки обозначены жирным шрифтом.

Критерий остановки алгоритма следующий [7, 8]: если, применяя пометки обоих типов, вершину n пометить не удалось, то полученный поток имеет максимальную величину.

Поток минимальной стоимости. Предположим, что задана сеть с пропускными способностями дуг c_{ij}. Пусть также для каждой дуги $(i; j)$ заданы число s_{ij}, интерпретируемое как затраты (например, затраты на перевозку единицы груза из вершины i в вершину j). Задача поиска потока минимальной стоимости заключается в нахождении для заданной величины φ суммарного потока ее распределения по дугам, минимизирующего сумму затрат. Общие методы решения задачи о потоке минимальной стоимости рассматриваются в [5, 7, 19].

Частным случаем задачи о потоке минимальной стоимости является ***транспортная задача***, в которой имеется *двудольный граф* (двудольным называется граф, множество вершин которого может

быть разбито на два непересекающихся подмножества, причем ребра (дуги) графа соединяют вершины только из разных подмножеств), представленный на рисунке 4: вершины сети разбиты на две группы – *m* поставщиков и *n* потребителей.

Известно [15], что граф является двудольным тогда и только тогда, когда он не содержит циклов нечетной длины, или когда в нем все простые циклы имеют четную длину (*теорема Кенига*).

Для поставщиков заданы имеющиеся у них количества единиц товара (груза и т.д.) a_i, $i = \overline{1,m}$, для потребителей – требуемые им количества единиц товара b_i, $i = \overline{1,n}$. Также известны затраты s_{ij} перевозки единицы товара от *i*-го поставщика *j*-му потребителю. Пусть задача является *замкнутой*, то есть $\sum_{i=1}^{m} a_i = \sum_{i=1}^{n} b_i$ – суммарное предложение равно суммарному спросу (вводя фиктивного поставщика или фиктивного потребителя любую незамкнутую задачу можно свести к замкнутой). Требуется определить потоки товаров от потребителей к поставщикам, минимизирующие суммарные затраты.

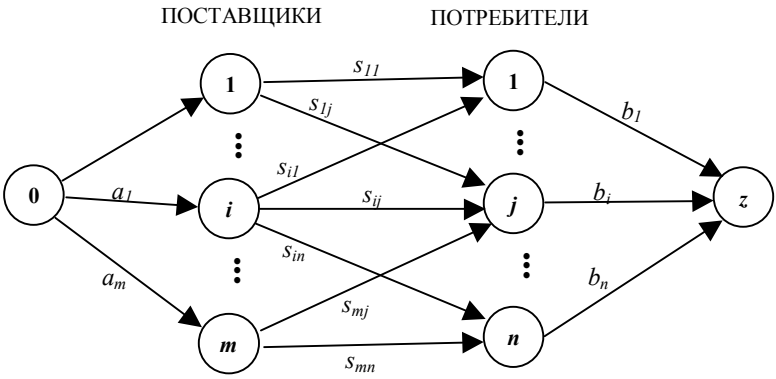

Рис. 4. Транспортная задача

Формально транспортную задачу можно записать в виде:

$$(1)\ \sum_{i=1}^{m}\ \sum_{j=1}^{n} x_{ij}\ s_{ij}\ \rightarrow \min_{\{x_{ij}\geq 0\}}$$

$$(2)\ \sum_{j=1}^{n} x_{ij}\ = a_i,\ i = \overline{1,m}$$

$$(3)\ \sum_{i=1}^{m} x_{ij}\ = b_j,\ j = \overline{1,n}\ .$$

Добавляя к двудольному графу вход «0» и выход «z» и соединяя вход и выход с остальными вершинами дугами с потоком $x_{0i} = a_i,\ ,\ i = \overline{1,m}\ ,\ x_{jz} = b_j,\ j = \overline{1,n}$, получаем задачу о потоке минимальной стоимости. Алгоритмы решения транспортной и двойственной к ней задач описаны в [7, 12].

Частным случаем транспортной задачи является ***задача о назначении***, заключающаяся в следующем: имеются n человек (работников), которые могут выполнять различные работы (занимать различные должности), число работ равно числу работников (введя фиктивные должности и/или фиктивные работы, всегда можно незамкнутую задачу привести к рассматриваемой замкнутой форме). Известны затраты s_{ij} на назначение i-го работника на j-ю должность (например, минимальная зарплата, за которую он согласится работать на этой должности). Требуется найти назначение работников на должности (каждого работника на одну и только одну должность), минимизирующее суммарные затраты (если s_{ij} интерпретируется как эффективность от работы i-го работника на j-ой должности, то оптимальное назначение должно максимизировать суммарную эффективность).

Формально задачу о назначении можно записать в виде (ср. с (1)-(3)):

$$(4)\ \sum_{i=1}^{n}\ \sum_{j=1}^{n} x_{ij}\ s_{ij}\ \rightarrow \min_{\{x_{ij}\in\{0;1\}\}}$$

$$(5)\ \sum_{j=1}^{n} x_{ij}\ = 1,\ i = \overline{1,n}$$

$$(6)\ \sum_{i=1}^{n} x_{ij}\ = 1,\ j = \overline{1,n}\ .$$

Известны множество методов решения задачи о назначении [7, 12]. Рассмотрим один из них на следующем примере.

Пусть имеются $n = 3$ работника и столько же работ. Матрица затрат имеет вид: $\begin{Vmatrix} 1 & 2 & 3 \\ 2 & 4 & 7 \\ 5 & 3 & 8 \end{Vmatrix}$.

Алгоритм 8.

Шаг 0. Назначаем каждого человека на самую дешевую для него работу (назначение выделено на рисунке 5 тонкими дугами), то есть положим $x_{ij}^0 = \begin{cases} 1, & \text{если } s_{ij} = \min\limits_k s_{ik} \\ 0, & \text{в противном случае} \end{cases}$. Если при этом назначение является допустимым (то есть все работы выполняются), то решение получено. Если имеется «дисбаланс», то есть не все работы выполняются ($\exists j_1$: $\sum\limits_{i=1}^{n} x_{ij_1}^0 > 1$), то переходим к следующему шагу.

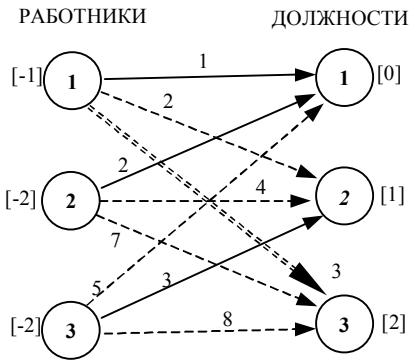

Рис. 5. Задача о назначении

Шаг k. Введем два подмножества множества дуг: $P_1 = \{(i; j) \mid x_{ij} = 1\}$, $P_2 = \{(i; j) \mid x_{ij} = 0\}$. Примем множество вершин-работ, на которых назначено несколько работников за вход сети, множество вершин-работ, которые не выполняются – за выход сети. Изменим направления дуг из множества P_1 на обратные и примем их длины

равными ($-s_{ij}$), длины дуг из множества P_2 примем равными s_{ij}. Найдем путь μ^k минимальной длины в полученной сети (потенциалы вершин, вычисляемые при нахождении кратчайшего пути в рассматриваемом примере, приведены в квадратных скобках). Далее полагаем $x_{ij}^k = \begin{cases} x_{ij}^{k-1}, & \text{если } (i;\ j) \notin \mu^k \\ 1 - x_{ij}^{k-1}, & \text{если } (i;\ j) \in \mu^k \end{cases}$.

В результате в рассматриваемом примере за один шаг получим оптимальное назначение, отличающееся от найденного на нулевом шаге тем, что первому работнику назначается третья работа (см. дугу, обозначенную двойными линиями на рисунке 5).

На каждом шаге число «дисбалансов» уменьшается на единицу. Следовательно, число шагов алгоритма не превышает числа «дисбалансов», которое конечно.

Аналогичным способом можно решить любую транспортную задачу (искать кратчайший путь из множества вершин, в которые доставили товара больше, чем требуется, во множество вершин, где товара не хватает).

Решение общего случая задачи о потоке минимальной стоимости основывается на рассмотрении двойственной задачи [7, 12].

5. Задачи календарно-сетевого планирования и управления

Рассмотрим *проект*, состоящий из набора *операций* (работ). Технологическая зависимость между операциями задается в виде сети (*сетевого графика*). При этом дуги сети соответствуют операциям, а вершины – событиям (моментам окончания одной или нескольких операций). Для каждой операции ($i; j$) задана ее продолжительность t_{ij}. Методы описания и исследования сетевых графиков изучаются в теории календарно-сетевого планирования и управления (КСПУ) [2, 3, 7, 8, 10, 11, 16].

Задача определения продолжительности проекта (управление временем). Легко видеть, что продолжительность проекта определяется путем максимальной длины, называемым *критическим путем*. Методы поиска пути максимальной длины описаны выше. Критический путь в сети на рисунке 6 выделен двойными дугами и равен 16.

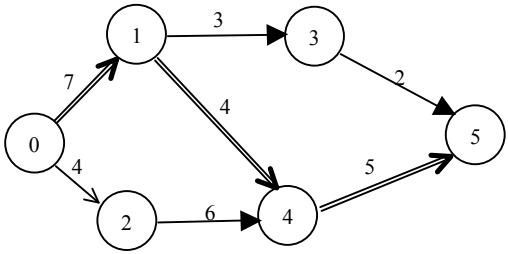

Рис. 6. Поиск критического пути

Операции, принадлежащие критическому пути, называются *критическими*. Остальные (некритические) операции имеют *резерв времени*, характеризуемый максимальной задержкой операции, при которой продолжительность проекта не изменяется. Критические операции имеют нулевой резерв. Приведем соответствующие формулы.

Алгоритм 9. Предположим, что выполнение комплекса операций (проекта) начинается в нулевой момент времени. Обозначим Q_0 – множество событий, не требующих выполнения ни одной из операций, то есть входы сети с правильной нумерацией; Q_i – множество событий, непосредственно предшествующих событию i, то есть множество вершин j сети, для которых существует дуга (j; i).

Положим

(1) $t_i^- = \max\limits_{j \in Q_0} t_{ji}$, $t_i^- = \max\limits_{j \in Q_i} (t_j^- + t_{ji})$.

Величина t_i^- называется *ранним моментом* (*временем*) *свершения* i-го события и характеризует время, раньше которого это событие произойти не может. *Длина критического пути*

(2) $T = \max\limits_i t_i^-$

определяется ранним временем свершения конечного события, то есть события, заключающегося в завершении всех операций.

Поздним моментом t_i^+ *свершения события* называется максимальное время его наступления, не изменяющее продолжительности проекта. Обозначим R_i – множество событий, непосредст-

венно следующих за событием i, то есть множество вершин j сети, для которых существует дуга $(i; j)$. Вычислим для каждой вершины-события i длину l_i максимального пути от этой вершины до выхода сети – события, заключающегося в завершении всего комплекса операций:

(3) $l_i = \max\limits_{j \in R_i} (l_j + t_{ij})$.

Положим $t_i^+ = T - l_i$, $i = \overline{1,n}$.

Для завершения проекта за время T необходимо и достаточно, чтобы событие i произошло не позднее момента t_i^+, $i = \overline{1,n}$.

Полным резервом Δt_i события i называется разность между его поздним и ранним моментами свершения, то есть

(4) $\Delta t_i = t_i^+ - t_i^-$, $i = \overline{1,n}$.

Очевидно, полный резерв критических событий (событий, принадлежащих критическому пути) равен нулю.

Задачи распределения ресурса на сетях удобно рассматривать, изображая операции вершинами сети, а зависимости – дугами (представления «операции-дуги, события-вершины» и «зависимости-дуги, операции-вершины» эквивалентны [10]). Пунктиром могут быть отражены ресурсные зависимости – когда для выполнения одних и тех же операций должны быть использованы одни и те же ресурсы. Примером могут являться сети, изображенные на рисунках 6 и 7. *Полным резервом операции* $(i; j)$ называется величина $\Delta_{ij} = t_{ij}^n - t_{ij}^p$, где t_{ij}^n – поздний срок начала (окончания) операции, а t_{ij}^p – ранний срок начала (окончания) операции.

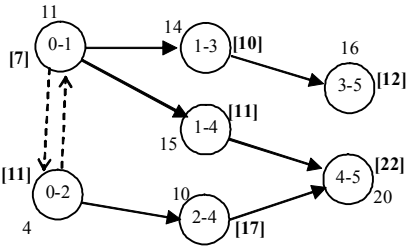

Рис. 7. Представление «операции-вершины» для сети рисунка 6

Для определения оптимального распределения ресурса необходимо найти критические пути для каждого из вариантов распределения ресурса и сравнить длины этих путей (в сети, приведенной на рисунке 7, существует общий для операций «0-1» и «0-2» ресурс; потенциалы вершин, соответствующие различным способам использования этого ресурса – сначала выполняется операция «0-1», затем «0-2» и наоборот, приведены на рисунке 7 соответственно в квадратных скобках и без скобок).

Универсальных эффективных точных методов решения задач распределения ресурсов на сетях не существует. В качестве частного случая, для которого существует простой алгоритм, приведем следующий пример.

В сети, изображенной на рисунке 8, для трех операций известны поздние времена окончания τ_i. Требуется определить очередность выполнения этих трех операций при условии, что все операции выполняются одной единицей ресурса и поэтому не могут выполняться одновременно.

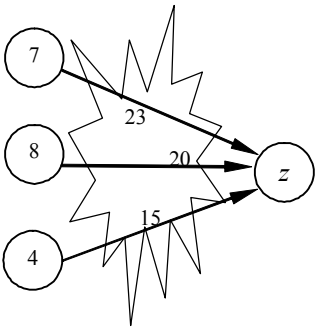

Рис. 8. Пример распределения ресурса

Легко показать, что в рассматриваемом примере оптимально выполнять первой операцию с минимальным τ_i.

Если для выполнения проекта выделено ограниченное количество ресурса, то возникает задача наилучшего его использования. Обозначим w_i – объем i-ой операции, $f_i(v_i)$ – скорость ее выполнения в зависимости от количества ресурса v_i. Предположим, что $f_i(\cdot)$ – непрерывная справа неубывающая функция, причем $f_i(0) = 0$. Если $v_i(t)$ – количество ресурса на i-ой операции в момент времени

t, то момент t_i ее окончания определяется как минимальное время, удовлетворяющее уравнению:

(5) $\int_0^{t_i} f_i(v_i(t))dt = w_i.$

Если количество ресурса, используемое при выполнении некоторой операции, не изменяется во времени, то говорят, что она выполняется с *постоянной интенсивностью*. Тогда продолжительность операции определяется выражением

(6) $t_i(v_i) = w_i / f_i(v_i)$.

В настоящее время общих алгоритмов поиска распределения ограниченных ресурсов между операциями, минимизирующего время завершения проекта, не существует. Поэтому рассмотрим несколько частных случаев.

Пусть все операции независимы и выполняются ресурсом одного вида, количество которого равно R, а $f_i(v_i)$ – непрерывные строго монотонные вогнутые функции. Тогда существует оптимальное решение, в котором каждая операция выполняется с постоянной интенсивностью и все операции заканчиваются одновременно в момент времени T, определяемый как минимальное время, удовлетворяющее следующему неравенству:

(7) $\sum_{i=1}^{n} f_i^{-1}(\frac{w_i}{T}) \le R$,

где $f_i^{-1}(\cdot)$ – функция, обратная функции $f_i(\cdot)$, $i = \overline{1,n}$ [7, 10].

Эвристические алгоритмы определения оптимального распределения ресурса для ряда случаев «невогнутых» функций интенсивности рассматриваются в [1-4].

Обширный класс задач КСПУ составляют *задачи агрегирования* – представления комплекса операций (проекта) в виде одной операции и исследования свойств таких представлений, для которых оптимизация в рамках агрегированного описания дает решение, оптимальное для исходного (детального) описания. Основные подходы к постановке и решению задач агрегирования рассматриваются в [1, 2, 10, 16].

Литература

1 Баркалов С.А., Бурков В.Н., Новиков Д.А., Шульженко Н.А. Модели и механизмы в управлении организационными системами. М.: Издательство «Тульский полиграфист», 2003. Том 1. – 560 с., Том 2 – 380 с., Том 3 – 205 с.

2 Баркалов С.А., Бурков В.Н., Гилязов Н.М. Методы агрегирования в управлении проектами. М.: ИПУ РАН, 1999. – 55 с.

3 Баркалов С.А., Бурков В.Н. Минимизация упущенной выгоды в задачах управления проектами. М.: ИПУ РАН, 2001. – 56 с.

4 Баркалов С.А., Бурков В.Н., Курочка П.Н., Образцов Н.Н. Задачи управления материально-техническим снабжением в рыночной экономике. М.: ИПУ РАН, 2000. – 58 с.

5 Берж К. Теория графов и ее применения. М.: Иностранная литература, 1962. – 319 с.

6 Бурков В.Н., Багатурова О.С., Иванова С.И. Оптимизация обменных производственных схем в условиях нестабильной экономики. М.: ИПУ РАН, 1996. – 48 с.

7 Бурков В.Н., Горгидзе И.А., Ловецкий С.Е. Прикладные задачи теории графов. Тбилиси: Мецниереба, 1974. – 234 с.

8 Бурков В.Н., Заложнев А.Ю., Новиков Д.А. Теория графов в управлении организационными системами. М.: Синтег, 2001. – 124 с.

9 Бурков В.Н., Зинченко В.Н., Сочнев С.В., Хулап Г.С. Механизмы обмена в экономике переходного периода. М.: ИПУ РАН, 1999. – 70 с.

10 Бурков В.Н., Ланда Б.Д., Ловецкий С.Е., Тейман А.И., Чернышев В.Н. Сетевые модели и задачи управления. М.: Советское радио, 1967. – 144 с.

11 Бурков В.Н., Новиков Д.А. Как управлять проектами. М.: Синтег, 1997. – 188 с.

12 Вагнер Г. Основы исследования операций. М.: Мир, 1972. Т. 1–4.

13 Воронин А.А., Мишин С.П. Оптимальные иерархические структуры. М.: ИПУ РАН, 2003. – 210 с.

14 Егоршин А.П. Управление персоналом. Н. Новгород: НИМБ, 1997. – 607 с.

15 Емеличев В.А,, Мельников О.И., Сарванов В.И., Тышкевич Р.И. Лекции по теории графов. М.: Наука, 1990. – 384 с.

16 Колосова Е.В., Новиков Д.А., Цветков А.В. Методика освоенного объема в оперативном управлении проектами. М.: Апостроф, 2001. – 156 с.

17 Коргин Н.А. Механизмы обмена в активных системах. М.: ИПУ РАН, 2003.

18 Новиков Д.А. Сетевые структуры и организационные системы. М.: ИПУ РАН, 2003. – 102 с.

19 Оре О. Теория графов. М.: Наука, 1968. – 352 с.