

Глава 4. ВВЕДЕНИЕ В ТЕОРИЮ СЕТЕВОГО ПЛАНИРОВАНИЯ

Пусть *проект* представляет собой список частично упорядоченных работ, направленных на достижение одной цели. Работа i предшествует работе j , если работа j не может быть начата раньше завершения работы i . Проект считается выполненным, если выполнены все работы списка. В любом проекте можно выделить подмножество *критических* работ, минимальное увеличение продолжительности выполнения которых приведет к увеличению времени выполнения всего проекта.

При анализе проекта необходимо определить, в частности, минимальное время его выполнения, а также найти узкие места.

Для анализа проекта в теории сетевого планирования используется сеть, отражающая отношения предшествования работ [5].

Пусть сеть $S = (X, U)$ – простой ориентированный ациклический граф с одним источником и одним стоком, где X – множество вершин, а U – множество дуг графа. Обозначим через $\mu(a, b)$ путь из вершины a в вершину b . Вершина i *предшествует* вершине k , если в сети S существует путь $\mu(i, k)$. Дуга (i, k) *предшествует* дуге (p, q) , если в сети S существует путь $\mu(k, p)$.

Говорят, что работа i *непосредственно предшествует* работе j , если i предшествует j и нет такой работы k , что i предшествует k и k предшествует j .

Сеть называется *взвешенной*, если ее вершинам или / и дугам приписаны числа – «веса». Длиной пути $\mu(i, k)$, обозначим ее $|\mu(i, k)|$, из вершины i в вершину k во взвешенной сети является сумма весов, включенных в путь элементов. Обозначим через $\mu^*(i, k)$ самый длинный путь из вершины i в вершину k .

Сетевая модель (сетевой график) – это представление списка работ проекта в виде взвешенной сети, которое позволяет адекватно отразить связи между работами проекта.

Существует два типа сетевых моделей, которые условно называют «работы-вершины» и «работы-дуги».

В сетевых моделях типа «работы-вершины» вершины являются образами работ, а дуги сети служат для представления отношения предшествования работ.

Для построения сети типа «работы-вершины» каждой работе проекта ставится в соответствие вершина, имеющая вес, равный длительности выполнения соответствующей работы. Из вершины i проводится дуга в вершину j , если работа i непосредственно предшествует работе j .

В моделях «работы-дуги» работы представлены дугами, а направление дуги отображает частичный порядок на множестве работ. Вершины в данном случае соответствуют *событиям*, которые можно охарактеризовать как моменты завершения одних работ и возможность начала выполнения других.

Для построения сети типа «работы-дуги» каждой работе проекта ставится в соответствие дуга, имеющая вес, равный длительности выполнения соответствующей работы. Из конечной вершины дуги p проводится дуга в начальную вершину дуги q , если работа p непосредственно предшествует работе q . Дуга, которая не имеет прообраза в множестве работ и введена для задания отношения предшествования, называется *фиктивной*. Нефиктивные дуги назовем *фактическими*.

Нельзя отдать предпочтение одному из способов представления проекта. Удобная форма представления работ, как правило, зависит от самого проекта и от особенностей используемого алгоритма. В рамках настоящего пособия, как и в работе [5], будем рассматривать сетевые модели типа «работы-дуги».

Далее предполагается, что каждой работе проекта поставлена в соответствие дуга сети. Каждой фактической дуге сети приписан вес, равный продолжительности соответствующей работы, а фиктивной дуге – нулевой вес.

4.1. Упрощение сети

Сети, описывающие крупномасштабные проекты, зачастую содержат большое число избыточных фиктивных дуг, которые можно удалить. Для этого используется процедура склейки вершин [4], которая описывается в данном разделе.

Далее будем использовать следующие обозначения:

$S = (X, U)$ – сеть с множеством вершин X и множеством дуг U ;

$i(j)$ – начальная вершина дуги j ;

$k(j)$ – конечная вершина дуги j ;

$\overline{U}$ – множество фактических дуг сети;

U_i – подмножество фактических дуг, принадлежащих путям из источника в вершину i ;

U^i – подмножество фактических дуг, принадлежащих путям из вершины i в сток.

Определение 4.1. Две сети S и S' с одинаковым множеством фактических дуг называются *эквивалентными*, если они представляют одно и то же отношение предшествования фактических дуг.

В этом случае в сети S' существует путь из вершины a в вершину b тогда и только тогда, когда существует путь из a в b в сети S .

Рассмотрим пару вершин u и v , для которых в сети S выполняются следующие два свойства:

1) не существует инцидентной им обеим *фактической* дуги;

2) не существует дуг $p, q \in \overline{U}$ таких, что:

– $i(p) = i(q), k(p) = u, k(q) = v$;

– или $k(p) = k(q), i(p) = u, i(q) = v$.

Результатом *склейки* вершин u и v в сети S , удовлетворяющих свойствам 1 и 2, является сеть S' , в которой вершина v удалена, а все дуги, которые были ей инцидентны в S , в S' инцидентны вершине u . Если в S' окажется несколько параллельных фиктивных дуг, то все они, кроме одной, удаляются.

Приведем без доказательства необходимое и достаточное условие эквивалентности сетей до и после склейки. Читатель, интересующийся доказательством теоремы, может обратиться к работе [4].

Теорема 4.1. Пусть вершины i и k удовлетворяют условиям 1 и 2. Сеть S' , полученная из S в результате склеивания вершин i, k и исключения k , эквивалентна S тогда и только тогда, когда $U_i = U_k$ или $U^i = U^k$.

4.2. Параметры сетевой модели

Будем считать, что каждой вершине сети приписан номер из множества $\{1, \dots, n\}$, а каждой дуге – номер из множества $\{1, \dots, m\}$. Более того, предположим, что источник имеет номер 1, а сток – номер n . Заметим, что, если первоначально в сетевой модели имеется несколько источников (стоков), можно ввести один фиктивный источник, который мы будем также называть *входом* и один

сток (*выход*), с которым нужно связать фиктивными дугами все источники (стоки).

Поскольку сетевая модель, как правило, является разреженной сетью, далее полагаем, что она представлена списком дуг, каждая из которых закодирована четырьмя числами:

- j – номер дуги;
- $i(j)$ – номер начальной вершины дуги j ;
- $k(j)$ – номер конечной вершины дуги j ;
- τ_j – длина дуги j (продолжительность выполнения соответствующей работы).

Определение 4.2. Ранним временем наступления события i называется величина $T_i^E = |\mu^*(I, i)|$.

Время $T = T_n^E$, которое, очевидно, соответствует максимальной длине пути из источника в сток, называется *критическим временем проекта*. Это минимально возможное время выполнения всего проекта. Путей из входа в выход сети максимальной длины может быть несколько. Любой путь $\mu^*(I, n)$, а также дуги (работы) и вершины (события), принадлежащие этому пути, также называются *критическими*.

Определение 4.3. Поздним временем наступления события i называется величина $T_i^L = T - |\mu^*(i, n)|$.

Если событие i наступит не позже момента T_i^L , то время выполнения всего проекта не увеличится.

Заметим, что событие i является *критическим* тогда и только тогда, когда $T_i^E = T_i^L$.

В общем случае не все работы проекта являются критическими. Продолжительность выполнения некоторых работ можно увеличить, а время выполнения проекта от этого не изменится. Максимальные величины, на которые можно увеличить время выполнения работы или задержать начало ее выполнения, без увеличения времени выполнения всего проекта, называют *резервами работ*.

Определение 4.4. Свободным резервом выполнения работы j называется величина $T_{k(j)}^E - T_{i(j)}^E - \tau_j$, которая равна максимальному увеличению продолжительности выполнения работы j , не меняющему ранние времена наступления всех событий.

Определение 4.5. Полным резервом выполнения работы j называется величина $T_{k(j)}^L - T_{i(j)}^E - \tau_j$. На такую величину можно увеличить продолжительность выполнения работы j , чтобы не изменилось критическое время проекта.

Очевидно, критические работы имеют нулевой свободный и полный резервы.

Определение 4.6. Ранг R_i события (вершины) i , равен максимальному количеству дуг в пути из входа l в вершину i .

Для нахождения рангов событий используется следующий алгоритм Форда.

Шаг 0. Для всех вершин $v = 1, \dots, n$ полагаем $r_v = 0$.

Пусть в результате $s - 1$ шагов найдены величины r_v .

Шаг s . Просматриваем последовательно работы-дуги $j = 1, \dots, m$ и пересчитываем величины $r_{k(j)} = \max\{r_{k(j)}, r_{i(j)} + 1\}$.

Если на каком-то шаге s ни одна из величин r_v , $v = 1, \dots, n$, не изменилась, то они являются рангами событий, и алгоритм останавливается. Иначе, полагаем $s = s + 1$ и повторяем шаг s .

Теорема 4.2. В результате выполнения s шагов алгоритма Форда для любой вершины v :

- 1) $r_v \leq R_v$;
- 2) $r_v = R_v$, если ранг события v не превосходит s .

Доказательство. Докажем утверждение 1 индукцией по рангу события. Для входа $r_1 = 0$, так как эта вершина не является конечной ни для одной дуги. Предположим неравенство справедливо для событий рангов $1, \dots, R$. Рассмотрим событие v , ранг которого $R_v = R + 1$. Пусть $k(j) = v$ и j – последняя работа, при рассмотрении которой изменилась величина r_v (т. е. произошло присваивание $r_v = r'_{i(j)} + 1$). Тогда, по предположению индукции,

$$r_v = r'_{i(j)} + 1 \leq r_{i(j)} + 1 \leq R_{i(j)} + 1 \leq R_v.$$

Докажем утверждение 2 индукцией по номеру шага алгоритма. На шаге 0 для единственной вершины ранга 0 имеем $r_1 = R_1 = 0$. Пусть в результате $s - 1$ шагов величины $r_l = R_l$ для всех событий l , ранги которых не превосходят $s - 1$, найдены. Рассмотрим событие v ранга s и работу j такую, что $k(j) = v$, $R_{i(j)} = s - 1$. По индукционному предположению, и утверждению 1 величина $r_{i(j)}$ не меняется на шаге s и равна $R_{i(j)}$. Следовательно, $r_v \geq r_{i(j)} + 1 = R_{i(j)} + 1 = s$. ♦

Отметим, что трудоемкость описанного выше алгоритма Форда равна $O(mn)$. Объем дополнительной памяти ограничен величиной $O(n)$.

Ранги событий позволяют осуществить, так называемую, правильную нумерацию вершин и правильную упорядоченность дуг, что дает возможность сократить трудоемкость вычисления ранних и поздних времен наступления событий.

Определение 4.7. Нумерация вершин называется *правильной*, если $i(j) < k(j)$ для всех дуг $j = 1, \dots, m$.

Зная ранги событий, легко осуществить правильную нумерацию вершин следующим образом:

- вход (вершина нулевого ранга) получает номер 1;
- вершинам ранга 1 приписываются номера 2, ..., n_1 ;
- вершины, имеющие ранг 2, нумеруются числами $n_1 + 1, \dots, n_2$;
- и т. д.
- окончательно, выход сети получает номер n .

Алгоритм Форда (нахождение ранних времен).

Шаг 0. Для всех вершин $v = 1, \dots, n$ полагаем $T_v^E = 0$.

Пусть в результате $s - 1$ шагов найдены значения T_v^E .

Шаг s. Просматриваем последовательно дуги $j = 1, \dots, m$ и пересчитываем величины $T_{k(j)}^E = \max\{T_{k(j)}^E, T_{i(j)}^E + \tau_j\}$.

Если на каком-то шаге s ранние времена событий не меняются, то алгоритм останавливается. В противном случае увеличиваем на единицу s и повторяем шаг s .

Определение 4.8. Если для любых дуг p и q , $p \in \mu(1, i(q))$ выполняется неравенство $p < q$, то говорят, что дуги сети *правильно упорядочены*.

Другими словами, если в проекте работа p предшествует работе q , то номер соответствующей ей дуги должен быть меньше номера дуги, соответствующей работе q .

Предположим, вершины занумерованы правильно. Для получения правильно упорядоченного списка дуг в этом случае достаточно воспользоваться номерами конечных вершин. Ниже приведено описание этой простой процедуры.

Шаг 0. Положим $v = 1$ и $n_v = 0$.

Шаг v . Дуги, входящие в вершину v , нумеруются числами $n_v + 1, \dots, n_v + m_v$, где $m_v = |\{u : (u, v) \in U\}|$. Если $v \neq n$, то полагаем $v = v + 1$, $n_v = n_v + m_v$ и повторяем шаг v . Иначе, процедура заканчивается.

Теорема 4.3. Если дуги правильно упорядочены, алгоритм Форда находит ранние времена наступления событий T_v^E за один просмотр дуг в порядке их нумерации $j = 1, \dots, m$.

Доказательство. Пусть дуги правильно упорядочены и величины T_v , $v = 1, \dots, n$ являются ранними временами, вычисленными на первом шаге алгоритма Форда. Предположим, что на втором шаге алгоритм не остановился. Значит, существует вершина v , у которой раннее время, полученное на первом шаге T'_v , меньше раннего времени T''_v , полученного на втором шаге. Пусть дуга j является первой в списке, когда величина $T_v = T_{k(j)}$ изменилась на втором шаге алгоритма. Тогда для вершины $i(j)$ раннее время также изменилось на втором шаге, т. е. $T'_{i(j)} < T''_{i(j)}$. Но дуги, заканчивающиеся в вершине $i(j)$, имеют номера меньше j . Значит, должно выполняться равенство $T'_{i(j)} = T_{i(j)}$ и изменение $T_{i(j)}$ на втором шаге алгоритма (до просмотра дуги j) противоречит выбору j . ♦

Алгоритм Форда (вычисление поздних времен).

Шаг 0. Для всех вершин $v = 1, \dots, n$ полагаем $T_v^L = T = T_n^E$.

Пусть на $s - 1$ шаге найдены величины T_v^L .

Шаг s . Просматриваем дуги $j = 1, \dots, m$ и пересчитываем величины $T_{i(j)}^L = \min\{T_{i(j)}^L, T_{k(j)}^L - \tau_j\}$.

Если все поздние времена не меняются, то алгоритм останавливается. Иначе полагаем $s = s + 1$ и повторяем шаг s .

Теорема 4.4. В случае правильной упорядоченности дуг поздние времена T_v^L находятся с помощью алгоритма Форда за один просмотр дуг в обратном порядке $j = m, \dots, 1$.

Когда ранние и поздние времена наступления всех событий известны, другие параметры (характеристики) сети, такие как критические события и работы, а также свободный и полный резервы выполнения работ, легко находятся за один просмотр списка дуг.

В данной главе рассмотрена простейшая модель сетевого планирования, которая позволяет определить характеристики проекта с полиномиальной трудоемкостью. На практике выполнение каждой работы проекта сопряжено с потреблением определенных ресурсов (финансов, сырья, материалов, рабочей силы и др.). В случае ограниченности ресурсов, задача сетевого планирования становится $\mathcal{NP}$ -трудной. С некоторыми подходами к решению такой задачи можно ознакомиться, например, в работе [5].