

# Теория расписаний

**Минимизация  
приоритето-порождающих функций**

# Минимизация приоритето-порождающих функций

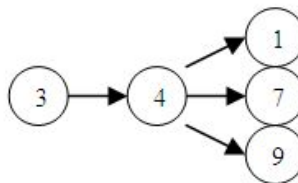
**Задача 1/out — tree/  $\sum C_j$**

**Решить задачу 1/out — tree/  $\sum C_j$ , в которой имеется 10 требований. Требование 3 предшествует требованию 4, которое, в свою очередь, предшествует требованиям 1, 7 и 9.**

**Длительности обслуживания  $p_j$  заданы в таблице:**

$j$	1	2	3	4	5	6	7	8	9	10
$p_j$	4	2	3	5	7	4	1	2	9	4

Для задачи 1//  $\sum C_j$  решением было бы расписание (7, {2, 8}, 3, {1, 6, 10}, 4, 5, 9). Однако это расписание нарушает отношения предшествования:



# Минимизация приоритето-порождающих функций

*Обозначим*

$\Pi_r$  - множество всех перестановок  $\pi_r = (i_1, \dots, i_r)$   
элементов множества  $N = \{1, \dots, n\}$ ,  $r = 1, \dots, n$

$$\Pi_0 = \{\pi_0\} = \{(\emptyset)\}$$

$$\Pi = \bigcup_{r=0}^n \Pi_r$$

где  $\cup$  – операция объединения множеств.

# Минимизация приоритето-порождающих функций

**Функция  $F(\pi)$ , определенная на множестве  $\Pi_n$  называется приоритето-порождающей (ППФ), если существует функция  $\omega(\pi)$ ,  $\pi \in \Pi$ , называемая функцией приоритета, которая обладает следующими свойствами:**

для любых перестановок  $\pi = (\pi^1, \pi^a, \pi^b, \pi^2) \in \Pi_n$  и  $\pi' = (\pi^1, \pi^b, \pi^a, \pi^2) \in \Pi_n$

- из  $\omega(\pi^a) > \omega(\pi^b)$  следует  $F(\pi) \leq F(\pi')$  и
- из  $\omega(\pi^a) = \omega(\pi^b)$  следует  $F(\pi) = F(\pi')$ .

# Минимизация приоритето-порождающих функций

*Множество  $N$  является частично упорядоченным, если задано отношение предшествования (бинарное, транзитивное, антирефлексивное отношение), представленное графом редукции этого отношения  $G = (N, U)$ .*

*Граф  $G$  называется графом редукции отношения предшествования, если он получен из графа отношения частичного порядка путем удаления всех транзитивных дуг.*

# Минимизация приоритето-порождающих функций

*Многие задачи построения оптимальных расписаний сводятся к минимизации ППФ на частично упорядоченных множествах требований.*

*Отношения предшествования присутствуют в задачах, где некоторые операции используют результаты других (предшествующих) операций.*

# Примеры приоритето-порождающих функций

Можно доказать, что:

для задачи  $1/prec/\Sigma C_j$  целевая функция является ППФ  
с функцией приоритета

$$\omega(\pi) = |\{\pi\}|/P(\pi)$$

где  $P(\pi) = \Sigma p_j$ ,

для задачи  $1/prec/\Sigma w_j C_j$  целевая функция является  
ППФ с функцией приоритета

$$\omega(\pi) = W(\pi)/P(\pi),$$

где  $W(\pi) = \Sigma w_j$ .

# Методы минимизации приоритето-порождающих функций на частично упорядоченных множествах

Пусть задано частично упорядоченное множество  $N$  с графом редукции отношения частичного порядка  $G = (N, U)$ .

Задача состоит в минимизации  $F(\pi)$ ,  $\pi \in \Pi_n(G)$ , где  $\Pi_n(G)$  - множество всех перестановок элементов множества  $N$ , допустимых относительно  $G$ .



# Методы минимизации приоритето-порождающих функций на частично упорядоченных множествах

**Введем операции** над бесконтурными орграфами, не содержащими транзитивных дуг (в т.ч. графами редукции отношения частичного порядка):

- **Преобразование I -  $[t, s]$  отождествления вершин  $t$  и  $s$ :** замена вершин  $t$  и  $s$  одной составной вершиной  $[t, s]$ .

Все входящие (исходящие) дуги в вершины  $t$  и  $s$  заменяются на входящие (исходящие) дуги в составную вершину. Удаляются появившиеся транзитивные дуги.

- **Преобразование II -  $(s, t)$  добавления дуги  $(s, t)$ :** добавление дуги  $(s, t)$  с последующим удалением транзитивных дуг.

## Методы минимизации приоритето-порождающих функций на частично упорядоченных множествах

Цепь  $(i_1, \dots, i_k)$ , где компоненты  $i_j$  являются составными вершинами, называется  $\omega$ -цепью, если  $\omega(i_l) \geq \omega(i_{l+1})$ ,  $l = 1, \dots, k - 1$ .

Если  $G$  представляет собой  $\omega$ -цепь, то перестановка  $(i_1, \dots, i_k)$  является оптимальной.

Если граф  $G$  – лес, то существует последовательность преобразований  $I$  и  $II$ , переводящая  $G$  в  $\omega$ -цепь.

# Алгоритм минимизации ППФ на частично упорядоченных множествах

Задача *1/out — tree/ F*, где  $F$  – ППФ.

Алгоритм минимизации ППФ на множестве  $\Pi_n(\mathbf{G})$ , где  $\mathbf{G}$  - набор выходящих деревьев :

- 1. Вычисляем **приоритеты** не имеющих потомков (*висячих*) вершин.
- 2. Если  $\mathbf{G}$  не есть набор изолированных вершин, то находим в  $\mathbf{G}$  вершину  $i_0$ , называемую **опорной**, все **прямые потомки которой** являются висячими.

Пусть этим потомкам соответствуют  $\omega$ -цепи  $C_1, \dots, C_r$

Построим  $\omega$ -цепь  $(i_1, \dots, i_r)$ , упорядочив все (составные) вершины цепей  $C_1, \dots, C_r$  по невозрастанию приоритетов.

## Алгоритм минимизации ППФ на частично упорядоченных множествах (продолжение)

Построим цепь  $(i_0, i_1, \dots, i_\nu)$ .

- Если  $\omega(i_0) > \omega(i_1)$ , то цепь  $(i_0, i_1, \dots, i_\nu)$  является  $\omega$ -цепью.
- Если  $\omega(i_0) \leq \omega(i_1)$ , то объединяем  $i_0$  и  $i_1$  в составную вершину  $[i_0, i_1]$ .

Далее сравниваем  $\omega(i_0, i_1)$  и  $\omega(i_2)$  и, в случае необходимости, объединяем  $[i_0, i_1]$  и  $i_2$  и т.д.

Процесс продолжается до тех пор, пока цепь  $(i_0, i_1, \dots, i_\nu)$  не будет преобразована в некоторую  $\omega$ -цепь  $\mathbf{C}^0 = ([i_0, i_1, \dots, i_k], i_{k+1}, \dots, i_\nu)$ .

Удаляем из  $\mathbf{G}$  всех потомков вершины  $i_0$  и ставим ей в соответствие  $\omega$ -цепь  $\mathbf{C}^0$ .

## Алгоритм минимизации ППФ на частично упорядоченных множествах (продолжение)

- 3. Повторяем описанный процесс до тех пор, пока не будет построен граф, состоящий из **изолированных вершин**.

*Последовательность (составных) вершин соответствующих  $\omega$ -цепям, в которой вершины упорядочены по невозрастанию приоритетов, является **оптимальным решением задачи**.*

## Алгоритм минимизации ППФ на частично упорядоченных множествах (продолжение)

В случае, когда граф  $G$  – входящее дерево, в качестве опорной выбирается вершина  $i_0$ , все непосредственные предшественники которой  $i_1, \dots, i_v$  не имеют предшественников.

Формируется цепь  $(i_1, \dots, i_v, i_0)$ . Она преобразуется в  $\omega$ -цепь путем сравнения  $\omega(i_0)$  и  $\omega(i_v)$ . Составная вершина  $[i_v, i_0]$  образуется, если  $\omega(i_v) \leq \omega(i_0)$ .

Далее процесс аналогичен случаю выходящего дерева.

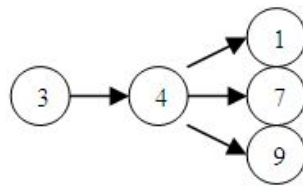
# Пример реализации алгоритма.

**Задача 1/out — tree/  $\sum C_j$**

**Решить задачу 1/out — tree/  $\sum C_j$** , в которой имеется **10** требований. Требование **3** предшествует требованию **4**, которое, в свою очередь, предшествует требованиям **1, 7 и 9**. Длительности обслуживания  $p_j$  заданы в таблице:

$j$	1	2	3	4	5	6	7	8	9	10
$p_j$	4	2	3	5	7	4	1	2	9	4

Граф отношений предшествования:



# Пример реализации алгоритма (продолжение).

1. Вычислим значение функции приоритета для  
висячих вершин.

Функция приоритета:

$$\omega(\pi) = |\{\pi\}|/P(\pi) \text{ где } P(\pi) = \sum p_j,$$

$j$	1	2	3	4	5	6	7	8	9	10
$\omega$	1/4	1/2			1/7	1/4	1	1/2	1/9	1/4



## Пример реализации алгоритма (продолжение).

*2a. Опорной вершиной является вершина 4, все прямые потомки которой 1, 7 и 9 являются висячими.*

*$\omega$ -цепь для потомков вершины 4: (7, 1, 9), поскольку значения функции  $\omega$  вершин равны, соответственно, (1, 1/4, 1/9)*

## Пример реализации алгоритма (продолжение).

2б. Цепь  $(4, 7, 1, 9)$  не является  $\omega$ -цепью, поскольку значения функции  $\omega$  вершины 4 равно  $1/5 < 1$ .

Объединяем вершины 4 и 7 в составную вершину  $[4, 7]$ .

Приоритет составной вершины  $[4, 7]$  равен  $2/(5+1) = 1/3$ . Цепь  $([4, 7], 1, 9)$  является  $\omega$ -цепью, т.к.  $1/3 > 1/4$ .

Удаляем из графа  $G$  всех потомков вершины 4 и ставим ей в соответствие  $\omega$ -цепь.

## Пример реализации алгоритма (продолжение).

***За.*** Опорной вершиной является вершина 3,  
 ***$\omega$ -цепь*** для потомков вершины 3: ***([4, 7], 1, 9)***

## Пример реализации алгоритма (продолжение).

**36.** Цепь  $(3, [4, 7], 1, 9)$  не является  $\omega$ -цепью поскольку значения функции  $\omega$  вершины 3 равно  $1/3=1/3$ .

Объединяем вершины 3 и  $[4, 7]$  в составную вершину  $[3, 4, 7]$ . Приоритет составной вершины  $[3, 4, 7]$  равен  $3/(3+5+1)=1/3$ .

Цепь  $([3, 4, 7], 1, 9)$  является  $\omega$ -цепью, т.к.  $1/3 > 1/4$ .

Удаляем из графа  $G$  всех потомков вершины 3 и ставим ей в соответствие  $\omega$ -цепь.

## Пример реализации алгоритма (продолжение).


4. Построен граф, состоящий из изолированных вершин.

Последовательность (составных) вершин соответствующих  $\omega$ -цепям, в которой вершины упорядочены по невозрастанию приоритетов ( $\omega$ ):

$(\{2, 8\}, 3, 4, 7, \{1, 6, 10\}, 5, 9)$ .

Данная последовательность является оптимальной.

Ответ:  $(\{2, 8\}, 3, 4, 7, \{1, 6, 10\}, 5, 9)$ , значение целевой функции равно 174.



$j$	{2	8}	3	4	7	{1	6	10}	5	9	
$p$	2	2	3	5	1	4	4	4	7	9	$\sum C_j =$
$C_j$	2	4	7	12	13	17	21	25	32	41	174