

РОССИЙСКАЯ АКАДЕМИЯ НАУК  
ВЫЧИСЛИТЕЛЬНЫЙ ЦЕНТР  
им. А.А. Дородницына

А.А. ЛАЗАРЕВ, Е.Р. ГАФАРОВ

ТЕОРИЯ РАСПИСАНИЙ.  
МИНИМИЗАЦИЯ СУММАРНОГО  
ЗАПАЗДЫВАНИЯ ДЛЯ ОДНОГО  
ПРИБОРА.

УДК 519.854.2

Ответственный редактор  
доктор физ.-матем. наук В.К. Леонтьев

Фундаментальными задачами теории расписаний для одного прибора являются задачи с критериями минимизации суммарного запаздывания и минимизация максимального временного смещения. В данной работе приводится достаточно полное исследование NP-трудной в обычном смысле задачи минимизация суммарного запаздывания для одного прибора  $1 \parallel \sum T_j$ .

Рецензенты: И.Х. Сигал,  
В.Н. Бурков

Научное издание  
©Вычислительный центр им. А.А. Дородницына  
Российской академии наук, 2006

## Введение

Данное издание является продолжением серии книг, посвященных проблемам теории расписаний, инициатором который был академик АН Белоруссии В.С. Танаев [1–4]. Фундаментальными задачами теории расписаний для одного прибора являются задачи с критериями минимизации суммарного запаздывания и минимизация максимального временного смещения. В данной работе приводится достаточно полное исследование NP-трудной в обычном смысле задачи минимизация суммарного запаздывания для одного прибора  $1 \parallel \sum T_j$  и ее взаимосвязь с задачей Разбиения.

Выделен ряд полиномиально и псевдополиномиально разрешимых случаев. При исследовании данной задачи были использованы как стандартные методы (динамическое программирование), так и методы, учитывающие специфические особенности задачи. Наряду с точными применялись популярные сегодня метаэвристические подходы (алгоритм "муравьиные колонии"). Кроме стандартного для теории расписаний перестановочного приема для разработки методов применялся метод локального поиска. Авторы признательны профессорам В.К. Леонтьеву, И.Х. Сигалу и В.Н. Буркову, взявшим на себя труд редактирования и рецензирования данной работы. Также хочется выразить признательность А.Г. Кваратцхелия за помощь в работе и обсуждение результатов.

А.А. Лазарев, Е.Р. Гафаров.

Февраль 2006 г.

## 1. Постановка задачи

Необходимо обслужить  $n$  требований на одном приборе. Прерывания при обслуживании и обслуживание более одного требования в любой момент времени запрещены. Для требования  $j \in N = \{1, 2, \dots, n\}$  заданы продолжительность обслуживания  $p_j > 0$  и директивный срок окончания обслуживания  $d_j$ , где  $N$  – множество требований, которые необходимо обслужить. Задан момент освобождения прибора  $t_0$ , с которого прибор готов начать обслуживание требований. Все требования поступают на обслуживание одновременно в момент времени  $t_0$ . Расписание обслуживания требований  $\pi$  строится с момента времени  $t_0$  и однозначно задается перестановкой элементов множества  $N$ .

Требуется построить расписание  $\pi^*$  обслуживания требований множества  $N$ , при котором достигается минимум функции  $F(\pi, t_0) = \sum_{j=1}^n \max\{0, c_j(\pi) - d_j\}$ , где  $c_j(\pi)$  – момент завершения обслуживания требования  $j$  при расписании  $\pi$ . Пусть  $\pi = (j_1, j_2, \dots, j_n)$ , тогда  $c_{j_1}(\pi) = t_0 + p_{j_1}$  и  $c_{j_k}(\pi) = c_{j_{k-1}}(\pi) + p_{j_k}$  для  $k = 2, 3, \dots, n$ . Величина  $T_j(\pi, t_0) = \max\{0, c_j(\pi) - d_j\}$  называется *запаздыванием* требования  $j$  при расписании  $\pi$ , а  $F(\pi, t_0)$  – *суммарным запаздыванием* требований множества  $N$  при расписании  $\pi$ , построенном с момента времени  $t_0$ . В случае, когда  $t_0 = 0$ , будем обозначать  $T_j(\pi)$  и  $F(\pi)$ .

Исследуемая проблема является NP-трудной в обычном смысле [5]. Е.Л. Лаулер [6] предложил псевдополиномиальный алгоритм решения общего случая проблемы трудоемкости  $O(n^4 \sum p_j)$ . В. Шварц и др. построили [7, 8] алгоритмы решения проблемы, которые были протестированы для примеров  $n < 600$  (тестовые примеры С.Н. Поттса и Л.Н. Ван Вассенхова [9]). Исследование приближенных алгоритмов решения

проблемы было проведено в работе [10], где построены примеры, на которых известные приближенные алгоритмы находят решение с относительной погрешностью порядка размерности примера  $n$ .

Во разд. 2 описываются известные правила сокращения перебора. Приводится **алгоритм А**, использующий при построении расписания правила исключения 1-3 [7, 11].

Частный случай задачи и алгоритмы его решения рассмотрены в разд. 3. В основе приведенных алгоритмов лежит правило разбиения исходного множества требований на подмножества. Для каждого подмножества рекурсивной процедурой находится частичное оптимальное расписание.

В разд. 4 исследуются два NP - трудных случая задачи – канонические примеры DL [5] и LG. Приводится доказательство NP - трудности частного случая **В-1**. Предложен алгоритм решения канонических примеров трудоемкости  $O(n \sum p_j)$  операций.

Показано, что известные алгоритмы [7, 8, 11, 13], для которых не получена оценка трудоемкости, имеют экспоненциальную трудоемкость для некоторых частных случаев задачи  $1 \parallel \sum T_j$ .

Свойства случая **В-1** и интерполяционный подход к решению исследуемой проблемы рассматривается в разд. 6.

В разд. 7 приводится сравнительный анализ эффективности **Гибридного алгоритма** и алгоритма "Муравьиные колонии".

В разд. 8 приводятся экспериментальная оценка количества точек изменения расписаний и оценка трудоемкости **алгоритма В-1 модифицированный**.

В разд. 9 представлена новая методика поиска "сложных" примеров.

## 2. Свойства задачи $1||\sum T_j$

**Определения.** Расписание  $\pi = (j_1, j_2, \dots, j_n)$  будем называть *SPT-расписанием* (short processing time), если выполняется  $p_{j_k} \leq p_{j_{k+1}}$  (для  $p_{j_k} = p_{j_{k+1}}$  имеем  $d_{j_k} \leq d_{j_{k+1}}$ ),  $k = 1, 2, \dots, n-1$ . Расписание  $\pi$  называют *LPT-расписанием* (large processing time), если  $p_{j_k} \geq p_{j_{k+1}}$  (для  $p_{j_k} = p_{j_{k+1}}$  выполняется  $d_{j_k} \leq d_{j_{k+1}}$ ),  $k = 1, 2, \dots, n-1$ .

Расписание  $\pi = (j_1, j_2, \dots, j_n)$  будем называть *EDD-расписанием* (early due date), если  $d_{j_k} \leq d_{j_{k+1}}$  (для  $d_{j_k} = d_{j_{k+1}}$  выполняется  $p_{j_k} \leq p_{j_{k+1}}$ ),  $k = 1, 2, \dots, n-1$ .

Расписание  $\pi'$  называется *частичным расписанием*, если при нем обслуживается некоторое подмножество требований  $N' \subset N$ . Также будем обозначать через  $P(N') = \sum_{i \in N'} p_i$ . Подмножество требований  $N' \subset N$ , обслуживаемых при частичном расписании  $\pi'$  будем обозначать через  $\{\pi'\}$ . Для частичного расписания  $\pi'$  суммарную продолжительность обслуживания требований множества  $\{\pi'\}$  будем обозначать  $P(\pi') = \sum_{i \in \{\pi'\}} p_i$ .

Через  $x = \langle \{p_j, d_j\}_{j \in N}, t_0 \rangle$  обозначим пример проблемы, для которого требуется построить оптимальное расписание. Подпримером примера  $x$  будем называть пару  $\{N', t'\}$ , где  $N' \subseteq N$ ,  $N' \neq \emptyset$ , и  $t' \geq t_0$ .

Без ограничения общности будем предполагать, что требования множества  $N$  пронумерованы в порядке неубывания директивных сроков

$$d_1 \leq d_2 \leq \dots \leq d_n,$$

если  $d_k = d_{k+1}$ , то  $p_k \leq p_{k+1}$ .

Через  $j^*(N')$  обозначим требование с наибольшей продолжительностью обслуживания среди требований множества  $N' \subseteq N$ , если таких требований несколько, то выбирается требование с наибольшим директивным сроком,

т.е.  $j^*(N') = \arg \max_{j \in N'} \{d_j : p_j = \max_{i \in N'} p_i\}$ . Для сокращения записи вместо  $j^*(N')$  будем записывать  $j^*$ , если очевидно о каком множестве идет речь.

Рассмотрим пример (подпример) обслуживания требований множества  $N' \subseteq N$ ,  $N' = \{1, 2, \dots, n'\}$ , с момента времени  $t' \geq t_0$ . Множество  $L(N', t')$  есть множество всех индексов  $k \in \{1, \dots, n'\}$ ,  $k \geq j^*(N')$ , таких что:

$$(a) \quad t' + \sum_{j=1}^k p_j < d_{k+1} \quad (\text{правило исключения 1 [8, 11]})$$

и

$$(b) \quad d_j + p_j \leq t' + \sum_{j=1}^k p_j, \quad \text{для всех } j = \overline{j^*(N') + 1, k}$$

(правила исключения 2, 3 [8, 11]),

где  $d_{n'+1} := +\infty$ .

**Теорема 1.** [11] Для любого примера (подпримера)  $\langle \{p_j, d_j\}_{j \in N}, t_0 \rangle$  множество  $L(N, t_0)$  не пусто.  $\square$

**Лемма 1.** [6, 9, 11] Для любого примера  $\langle \{p_j, d_j\}_{j \in N}, t_0 \rangle$  существует оптимальное расписание  $\pi^*$  обслуживания требований множества  $N$ , при котором  $(j \rightarrow j^*)_{\pi^*}$  для всех требований  $j \in \{1, 2, \dots, k\} \setminus \{j^*\}$  и  $(j^* \rightarrow j)_{\pi^*}$  для всех требований  $j \in \{k+1, \dots, n\}$  для некоторого  $k \in L(N, t_0)$ .  $\square$

Через  $(i \rightarrow j)_{\pi}$  обозначают, что требование  $i$  обслуживается раньше требования  $j$  при расписании  $\pi$ .

Опишем алгоритм нахождения оптимального расписания на основе использования правил исключения 1 – 3.

### Процедура ProcL ( $N, t$ )

0. Дан пример  $\{N, t\}$  с множеством требований  $N = \{j_1, j_2, \dots, j_n\}$  и моментом начала обслуживания  $t$ ,  $d_{j_1} \leq d_{j_2} \leq \dots \leq d_{j_n}$ ;
1. **IF**  $N = \emptyset$  **THEN**  $\pi^* :=$  пустое расписание, **GOTO** 6.;
2. Найдем требование  $j^*(N, t)$  из множества  $N$  с момента времени  $t$ ;
3. Найдем множество  $L(N, t)$  для требования  $j^*$ ;
4. **FOR ALL**  $k \in L(N, t)$  **DO**:  
$$\pi_k := (\mathbf{ProcL}(N', t'), j^*, \mathbf{ProcL}(N'', t'')),$$
 где  
$$N' := \{j_1, \dots, j_k\} \setminus \{j^*\}, t' := t,$$
$$N'' := \{j_{k+1}, \dots, j_n\}, t'' := t + \sum_{i=1}^k p_{j_i};$$
5.  $\pi^* := \arg \min_{k \in L(N, t)} \{F(\pi_k, t)\}$ ;
6. **RETURN**  $\pi^*$ .

### Алгоритм A

$$\pi^* := \mathbf{ProcL}(N, t_0).$$

Пусть  $N = \{j_1, \dots, j_n\}$ ,  $d_{j_1} \leq \dots \leq d_{j_n}$ . Модифицированное EDD-расписание, при котором требование  $j^*$  обслуживается  $k$ -м по порядку, обозначим через  $\pi^k = (j_1, \dots, j_{m-1}, j_{m+1}, \dots, j_k, j^*, j_{k+1}, \dots, j_n)$ , требование  $j^* = j_m$ ,  $m \leq k$ .

**Лемма 2. Правило исключения 4** [7, 13]. Если  $F(\pi^k) > F(\pi^{k+1})$  или  $F(\pi^k) \geq F(\pi^i)$  для требования  $j^* \leq i < k$ , то позиция  $k$  исключается из списка "подходящих" позиций  $L(N', t')$  для задачи  $\langle \{p_j, d_j\}_{j \in N'}, t' \rangle$ , если для множества  $|L(N', t')| > 1$ .  $\square$

Пусть  $B_j$  – список требований, обслуживающихся перед требованием  $j$ , а  $A_j$  – список требований обслуживающихся после требования  $j$  при любом оптимальном расписании. Множества  $B_j$  и  $A_j$  могут быть пустыми одновременно.

Определим  $E_j = P(B_j) + p_j + t'$ ,  $L_j = P(N' \setminus A_j) + t'$  как наименьшее и наибольшее значения момента окончания  $c_j$  при любом оптимальном расписании обслуживания требований множества  $N'$  с момента времени  $t'$ .

**Лемма 3. Условия Эммонса** [16]. Существует оптимальное расписание  $\pi^*$ , при котором

1.  $i$  предшествует  $j$ ,  $(i \rightarrow j)_{\pi^*}$ , если  $d_i \leq \max(E_j, d_j)$  и  $p_i \leq p_j$ ;
2.  $j$  предшествует  $i$ ,  $(j \rightarrow i)_{\pi^*}$ , если для этих требований выполняется  $d_i + p_i \geq L_j$  и  $d_i > \max(E_j, d_j)$ ,  $p_i \leq p_j$ .  $\square$

**Лемма 4.** [6] Пусть  $\pi$  – некоторое оптимальное расписание и

$$\min\{d_j, c_j(\pi)\} \leq d'_j \leq \max\{d_j, c_j(\pi)\},$$

для всех  $j \in N$ . Тогда любое оптимальное расписание  $\pi'$  для модифицированного примера  $\langle \{p_j, d'_j\}_{j \in N}, t \rangle$  с директивными сроками  $d'_1, d'_2, \dots, d'_n$  является оптимальным и для исходного примера  $\langle \{p_j, d_j\}_{j \in N}, t \rangle$  с директивными сроками  $d_1, d_2, \dots, d_n$ .  $\square$

Предлагается искать оптимальное решение модифицированного примера, у которого  $p'_j = p_j$  и, кроме того,  $d'_j = \max\{E_j, d_j\}$ ,  $j \in N$ , [7].

### 3. Случай В проблемы $1||\sum T_j$

Исследуется частный случай проблемы, когда параметры требований удовлетворяют условиям:

$$\begin{cases} p_1 \geq p_2 \geq \dots \geq p_n, \\ d_1 \leq d_2 \leq \dots \leq d_n. \end{cases} \quad (1)$$

Мы полагаем, что классу примеров, параметры требований которых удовлетворяют (1), принадлежат "наиболее трудоемкие" для решения примеры. В частности, "плохие" (в оригинале, bad) примеры из работы [10] удовлетворяют ограничениям (1).

В данной работе мы предлагаем подход к решению проблемы, когда исходное множество требований разбивается на подмножества, а затем на основе этого разбиения строится оптимальное расписание.

#### Процедура разбиения множества требований

$N = \{j_1, \dots, j_n\}$ ,  $d_{j_1} \leq \dots \leq d_{j_n}$ , на подмножества

0.  $k := 1$ ,  $\alpha_k := j_1$ ;

1. FOR  $i = 2, 3, \dots, n$  DO:

    IF  $d_{j_i} - d_{\alpha_k} > p_{j_i}$  THEN

$\beta_k := j_{i-1}$ ;

$k := k + 1$ ;

$\alpha_k := j_i$ ;

2.  $\beta_k = j_n$ .

Данная процедура однозначно разбивает исходное множество  $N$  на  $k$  подмножеств,  $k \leq n$ , за  $O(n)$  операций. В результате работы процедуры будут выделены подмножества требований  $M_1, M_2, \dots, M_k$ , где  $M_i = \{\alpha_i, \alpha_i + 1, \dots, \beta_i\}$ ,

$\alpha_1 = j_1, \beta_k = j_n$ . При этом,  $M_1 \cup M_2 \cup \dots \cup M_k = N$  и  $M_i \cap M_j = \emptyset$ , если  $i \neq j$ .

Например, для множества из трех требований с параметрами:  $p_1 = 10, p_2 = 10, p_3 = 2, d_1 = 7, d_2 = 9, d_3 = 10$ , будет получено разбиение на подмножества  $M_1 = \{1, 2\}$  и  $M_2 = \{3\}$ .

Для произвольных примеров  $x_1 = \langle \{p_j, d_j\}_{j \in N}, t \rangle$  и  $x_2 = \langle \{p_j, d_j + C\}_{j \in N}, t + C \rangle$ , где  $C$  – константа, множества оптимальных расписаний и оптимальные значения целевой функции совпадают. Такие примеры  $x_1$  и  $x_2$  будем называть *эквивалентными примерами*.

В следующих подразделах предлагаются алгоритмы решения случая (1) в зависимости от значения  $k$  (количества подмножеств, выделенных процедурой разбиения).

### 3.1. Случай В-1

В случае  $k = 1$  параметры требований исходного примера удовлетворяют соотношениям

$$\begin{cases} p_1 \geq p_2 \geq \dots \geq p_n, \\ d_1 \leq d_2 \leq \dots \leq d_n, \\ d_n - d_1 \leq p_n, \end{cases} \quad (2)$$

т. е.  $d_1 \leq d_2 \leq \dots \leq d_n \leq d_1 + p_n$ .

Далее в работе будем использовать следующие обозначения. Пусть

$$d_j(t) = d_j - d_n + t - t_0, \quad j \in N, \quad (3)$$

при этом, с учетом (2), для любого  $t \in \mathbb{R}$  выполняется  $d_1(t) \leq d_2(t) \leq \dots \leq d_n(t)$ . Пусть  $\pi_l(t)$  и  $F_l(t)$  оптимальное расписание и соответствующее ему значение суммарного запаздывания для примера с подмножеством требований  $N_l = \{l, \dots, n\}$ , моментом начала обслуживания равным 0, и директивными сроками окончания обслуживания  $d_j(t)$ ,  $j = l, \dots, n$ ,  $l = n, \dots, 1$ . Напомним, что  $F(\pi, t_0)$  – суммарное запаздывание требований множества  $N$  при расписании  $\pi$ , построенном с момента времени  $t_0$ .

Согласно определению величин  $d_j(t)$  (3), пример  $\langle \{p_j, d_j(t)\}_{j \in N}, 0 \rangle$  эквивалентен примеру  $\langle \{p_j, d_j\}_{j \in N}, d_n - t + t_0 \rangle$ . Также пример  $\langle \{p_j, d_j(t)\}_{j \in N}, 0 \rangle$  при  $t = d_n$  совпадает с исходным примером  $\langle \{p_j, d_j\}_{j \in N}, t_0 \rangle$ .

Если  $t \leq t_0 + p_n$ , тогда для примера  $\langle \{p_j, d_j(t)\}_{j \in N}, 0 \rangle$  будем иметь

$$d_j(t) \leq d_n(t) = t - t_0 \leq p_n \leq p_j, \quad j \in N.$$

В этом случае при любом расписании, построенном с момента времени 0, все требования множества  $N$  будут запаздывать,

поэтому оптимальным будет SPT-расписание  $(n, n-1, \dots, 1)$ .  
 Если  $t \geq t_0 + \sum_{j=1}^n p_j$ , тогда для примера  $\langle \{p_j, d_j(t)\}_{j \in N}, 0 \rangle$ ,  
 учитывая  $p_n \geq d_n - d_1$ , будем иметь  $t + p_n \geq t_0 + \sum_{j=1}^n p_j + d_n - d_1$ .  
 Отсюда следует

$$\sum_{j=1}^n p_j - p_n \leq d_1 - d_n + t - t_0 = d_1(t) \leq d_j(t), \quad j \in N,$$

т.е., при любом расписании запаздывающим будет не более одного требования, которое обслуживается последним по порядку, поэтому EDD-расписание  $(1, 2, \dots, n)$  будет оптимальным.

Таким образом, при изменении  $t$  от  $t_0 + p_n$  до момента времени  $t_0 + \sum_{j=1}^n p_j$  оптимальное расписание для примера  $\langle \{p_j, d_j(t)\}_{j \in N}, 0 \rangle$  изменяется от  $(n, n-1, \dots, 1)$  до  $(1, 2, \dots, n)$ . Интересными являются вопросы: в каких точках происходит изменение оптимального расписания и как много таких точек.

Приведем алгоритм решения примеров, параметры требований которых удовлетворяют условиям (2).

**Алгоритм В-1** решения исходного примера  $\langle \{p_j, d_j\}_{j \in N}, t_0 \rangle$  в случае (2). Входные параметры алгоритма:  $p_j, d_j, j \in N, t_0$ .

0.  $\pi_n(t) := (n), F_n(t) := \max\{0, p_n + t_0 - t\}, \forall t$ ;

1. **FOR**  $l = n - 1, n - 2, \dots, 1$ :

$$\pi^1 := (l, \pi_{l+1}(t - p_l)), \pi^2 := (\pi_{l+1}(t), l);$$

$$F(\pi^1) := \max\{0, p_l - d_l(t)\} + F_{l+1}(t - p_l);$$

$$F(\pi^2) := F_{l+1}(t) + \max\{0, \sum_{j=l}^n p_j - d_l(t)\};$$

$$F_l(t) := \min\{F(\pi^1), F(\pi^2)\};$$

$$\pi_l(t) := \arg \min\{F(\pi^1), F(\pi^2)\}.$$

2. Алгоритм возвращает расписание  $\pi_1(d_n)$  и соответствующее ему значение суммарного запаздывания  $F_1(d_n)$ .

Шаги основного цикла выполняются для целых значений  $t \in [0, \sum p_j]$ . Для точек  $t - p_l < 0$  требования при оптимальном частичном расписании обслуживаются в SPT-порядке.

В алгоритме В-1 на каждом шаге для всех возможных точек  $t$  строятся расписания  $\pi_l(t)$  путем включения требования  $l$  на первое или последнее место в ранее построенные расписания  $\pi_{l+1}(t - p_l)$  или  $\pi_{l+1}(t)$ , соответственно, и выбирается наилучшее из двух получившихся расписаний. Далее доказывається, что такой способ построения позволяет получить оптимальное расписание для случая (2).

Будем обозначать через  $N_l = \{l, l + 1, \dots, n\}$ .

**Теорема 2.** Расписание  $\pi_1(d_n)$  является оптимальным для случая (2). **Алгоритм В-1** строит расписание  $\pi_1(d_n)$  за  $O(n \sum p_j)$  операций.

**Доказательство.** Сначала покажем, что для всех  $l \in N$  существует оптимальное расписание  $\pi^* = (\pi_1^*, \pi_l, \pi_2^*)$ , где  $\{\pi_l\} = N_l = \{l, \dots, n\}$ ,  $\{\pi_1^*\} \cup \{\pi_2^*\} = \{1, \dots, l\}$ .

Предположим, что для всех оптимальных расписаний  $\pi$  существует такое  $l \in N$ , что  $\pi = (\pi_1, j, \pi_2)$ , где требование  $j \notin N_l$ ,  $\{\pi_1\} \cap N_l \neq \emptyset$ ,  $\{\pi_2\} \cap N_l \neq \emptyset$ . Выберем два требования  $\alpha$  и  $\beta$  так, что  $\alpha \in \{\pi_1\}$ ,  $\beta \in \{\pi_2\}$ ,  $\alpha, \beta \in N_l$ .  $\pi_1 = (\pi_{11}, \alpha, \pi_{12})$  и  $\pi_2 = (\pi_{21}, \beta, \pi_{22})$ ,  $\{\pi_{22}\} \cap N_l = \emptyset$ ,  $\{\pi_{11}\} \cap N_l = \emptyset$ , тогда расписание  $\pi = (\pi_{11}, \alpha, \pi_{12}, j, \pi_{21}, \beta, \pi_{22})$ . Из (2) мы имеем  $p_1 \geq \dots \geq p_n$  и  $d_1 \leq \dots \leq d_n$ , поэтому для  $j \notin N_l$  будет выполняться  $p_j \geq p_\alpha$ ,  $p_j \geq p_\beta$  и  $d_j \leq d_\alpha$ ,  $d_j \leq d_\beta$ .

Рассмотрим следующие случаи.

**1)** Пусть  $c_j(\pi) \leq d_j$ . Из расписания  $\pi$  построим расписание  $\pi' = (\pi_{11}, \pi_{12}, j, \alpha, \pi_2)$ . При расписаниях  $\pi$  и  $\pi'$  выполняется  $c_\alpha(\pi') = c_j(\pi) \leq d_j \leq d_\alpha$ , а также для всех  $i \in \{\pi_{12}\}$  верно  $c_i(\pi') \leq c_i(\pi)$ . То есть требования  $j$  и  $\alpha$  не запаздывают при обоих расписаниях, поэтому  $F(\pi') \leq F(\pi)$ . После такого сдвига требования  $\alpha$  имеем  $c_j(\pi') < d_j$ , так как имеем  $p_\alpha > 0$ .

**2)** Пусть  $c_j(\pi) > d_j$  и  $c_j(\pi) \geq d_\beta$ . Обозначим  $A = c_j(\pi) - p_j$ ,  $B = \sum_{j \in \{\pi_{21}\}} p_j$ . Построим расписание  $\pi' = (\pi_1, \beta, \pi_{21}, j, \pi_{22})$ . При расписаниях  $\pi$  и  $\pi'$  выполняется.

$$F(\pi) = F(\pi_1) + A + p_j - d_j + F(\pi_{21}, A + p_j) + A + p_j + B + p_\beta - d_\beta + F(\pi_{22}, A + p_j + B + p_\beta).$$

$$F(\pi') = F(\pi_1) + \max\{0, A + p_\beta - d_\beta\} + F(\pi_{21}, A + p_\beta) + A + p_\beta + B + p_j - d_j + F(\pi_{22}, A + p_j + B + p_\beta).$$

Из неравенства  $p_j \geq p_\beta$  следует  $F(\pi_{21}, A + p_\beta) \leq F(\pi_{21}, A + p_j)$ . Тогда выполняется

$F(\pi') - F(\pi) \leq \max\{0, A + p_\beta - d_\beta\} - A - p_j + d_\beta \leq 0$ , т.е. будем иметь  $F(\pi') \leq F(\pi)$ .

**3)** Пусть  $c_j(\pi) > d_j$  и  $c_j(\pi) < d_\beta$ . Построим расписание  $\pi' = (\pi_{11}, \pi_{12}, j, \alpha, \pi_2)$ . При расписаниях  $\pi$  и  $\pi'$  выполняется

$$\begin{aligned} T_j(\pi) &= c_j(\pi) - d_j, & T_j(\pi') &= 0, \text{ т.к. } c_j(\pi') = c_j(\pi) - p_\alpha < \\ & & & d_\beta - p_\alpha \leq d_1 \leq d_j, \\ T_\alpha(\pi) &= 0, & T_\alpha(\pi') &= \max\{0, c_j(\pi) - d_\alpha\}, \end{aligned}$$

а также  $T_i(\pi) \geq T_i(\pi')$  для всех  $i \in N$ ,  $i \neq j, \alpha$ . Тогда верно  $F(\pi') - F(\pi) \leq \max\{0, c_j(\pi) - d_\alpha\} - c_j(\pi) + d_j \leq 0$ , т.е. получим  $F(\pi') \leq F(\pi)$ .

Мы показали, что для всех расписаний  $\pi = (\pi_1, j, \pi_2)$ , у которых  $j \notin N_l$ , и  $\{\pi_1\} \cap N_l \neq \emptyset$ ,  $\{\pi_2\} \cap N_l \neq \emptyset$ , для некоторого  $l$ ,  $l = 1, \dots, n$ , можно построить соответствующее расписание  $\pi'$ , что  $F(\pi') \leq F(\pi)$ . Операции преобразования расписания в случаях 1), 2) и 3) приводят к тому, что требование  $j \notin N_l$  "выталкивается" из группы требований с большими номерами (требования множества  $N_l$ ). За конечное число операций любое расписание  $\pi$  может быть приведено к расписанию  $\pi' = (\pi_1, \pi_l, \pi_2)$ , где  $\{\pi_l\} = N_l$  и, кроме того,  $\{\pi_1\} \cup \{\pi_2\} = \{1, \dots, l\}$ , причем  $F(\pi') \leq F(\pi)$ . Таким образом, существует оптимальное расписание  $\pi^*$ , при котором для всех  $l = n, n-1, \dots, 1$  и для всех  $j \in N \setminus N_l$  выполняется либо  $(j \rightarrow i)_{\pi^*}$ , либо  $(i \rightarrow j)_{\pi^*}$  для всех  $i \in N_l$ .

Покажем, что **алгоритм В-1** строит оптимальное расписание для случая (2). Оптимальное расписание для набора требований  $N_n = \{n\}$ , очевидно,  $\pi_n(t) = (n)$  для любых  $t$ . Для набора требований  $N_{n-1} = \{n, n-1\}$  оптимальным расписанием будет одно из двух расписаний  $(n-1, n)$  и  $(n, n-1)$ . Для каждого  $t$  в **алгоритме В-1** оба расписания просматриваются и выбирается наилучшее, которое и является оптимальным. Пусть на шаге  $(l+1)$  алгоритма для всех  $t$  построены расписания  $\pi_{l+1}(t)$ , которые

являются оптимальными для соответствующих примеров с множеством требований  $N_{l+1}$ . Согласно доказанному выше свойству, существует оптимальное расписание  $\pi_l(t)$ , при котором для всех  $t$  выполняется или  $(l \rightarrow j)_{\pi_l(t)}$ , или верно  $(j \rightarrow l)_{\pi_l(t)}$  для всех  $j \in N_{l+1}$ . Обе ситуации рассматриваются на каждом шаге алгоритма для каждого  $t$ . Поэтому расписания  $\pi_l(t)$  будут оптимальными для каждого  $t$ . Следовательно, оптимальным расписанием исходного примера будет  $\pi_1(d_n)$ .

Вычислим трудоемкость **алгоритма В-1**. Согласно алгоритму, при построении расписания  $\pi_l(t)$  в точке  $t$  мы обращаемся к расписаниям  $\pi_{l+1}(t)$  и  $\pi_{l+1}(t-p_l)$ , построенным на шаге  $(l+1)$  алгоритма. Вычислять расписания для точек  $t \leq t_0 + p_n$  и  $t \geq t_0 + \sum_{j=1}^n p_j$  не нужно поскольку, как было показано выше, в этих точках оптимальные расписания известны (SPT и EDD последовательности, соответственно) и могут быть построены до работы алгоритма. Поэтому для построения оптимального расписания исходного примера необходимо для каждого требования  $l$  просматривать точки  $t$  в интервале  $[t_0 + p_n, t_0 + \sum_{j=1}^n p_j]$ . Шаг алгоритма для фиксированных  $l$  и  $t$  выполняется за  $O(1)$  операций. Основной шаг алгоритма для фиксированного требования  $l$  выполняется за  $O(\sum p_j)$  операций, так как  $p_j \in \mathbb{Z}^+$ ,  $j \in N$ , и достаточно рассматривать только целочисленные точки  $t$  на указанном интервале. Следовательно, трудоемкость всего алгоритма составляет  $O(n \sum p_j)$  операций. Теорема доказана.  $\square$

При реализации **алгоритма В-1** необходимо учитывать тот факт, что величина  $d_n$  должна быть целочислена. Поэтому, если  $d_n \notin \mathbb{Z}$ , предлагается построить и решить с помощью **алгоритма В-1** следующий пример:  $\langle \{p_j, d'_j\}_{j \in N}, t'_0 \rangle$ ,  $d'_j = d_j - \Delta$ ,  $j \in N$ ,  $t'_0 = t_0 - \Delta$ , где  $\Delta = d_n - \lfloor d_n \rfloor$ , где  $\lfloor a \rfloor$  есть целая часть числа  $a$ . В

этом случае величина  $d'_n$  является целочисленной. Исходный и построенный примеры эквивалентны между собой, следовательно, расписание  $\pi_1(d'_n)$  является оптимальным для исходного примера  $\langle \{p_j, d_j\}_{j \in N}, t_0 \rangle$ .

### 3.2. Случай ВФ

Пусть для случая (2) в любом из  $n!$  расписаний запаздывает ровно  $m$  требований. Для этого случая предлагается **алгоритм ВФ** трудоемкости  $O(n^2)$  операций.

**Лемма 5.** Пусть для случая (2) известно, что при любом из  $n!$  расписаний запаздывает ровно  $m$  требований. Существует оптимальное расписание, при котором требование  $j^* = \operatorname{argmax}_{j \in N} \{mp_j + d_j\}$  запаздывает.

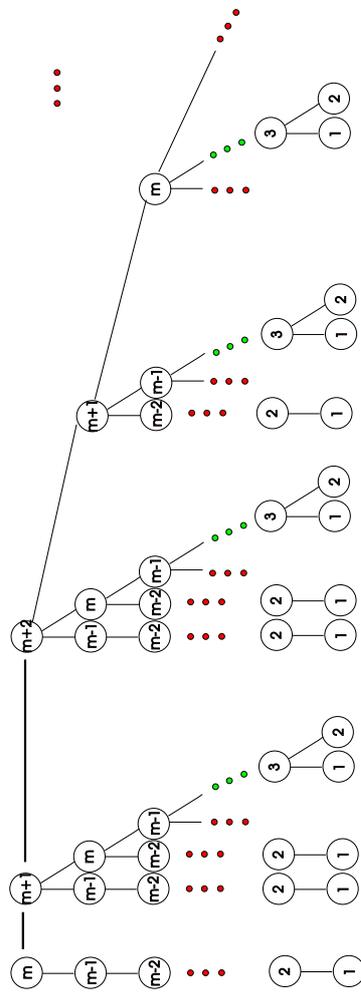
**Доказательство.** Пусть требование  $j^*$  не запаздывает в оптимальном расписании  $\pi = (\pi_1, j^*, \pi_2, j)$ . Рассмотрим расписание  $\pi' = (\pi_1, j, \pi_2, j^*)$ . Так как  $mp_{j^*} + d_{j^*} \geq mp_j + d_j$ , то выполняется  $F(\pi) \geq F(\pi')$ . Следовательно существует оптимальное расписание  $\pi'$ , при котором требование  $j^*$  запаздывает.  $\square$

**Лемма 6.** Пусть для случая (2) в любом из  $n!$  расписаний запаздывает ровно  $m$  требований, тогда в любом оптимальном расписании  $\pi^*$  запаздывающие требования упорядочены в конце расписания по правилу SPT. Истинность леммы устанавливается леммой 10.

Задача поиска оптимального расписания сводится к выбору  $m$  запаздывающих требований  $j_1, \dots, j_m$ . Учитывая, что запаздывающие требования упорядочены в SPT порядке, то первым запаздывающим требованием может быть только требование  $l = n - m + 1, \dots, n$ .

Обозначим через  $R$  множество частичных расписаний,  $WR$  – приоритет текущей "оптимальной" цепочки,  $TR$  – запаздывание требований текущей "оптимальной" цепочки,  $PR$  – продолжительность обслуживания цепочки.

**Полная схема вычислений**



**Сокращенная схема вычислений**

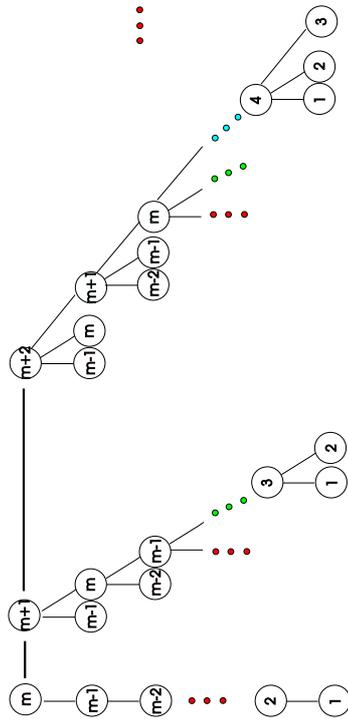


Рис. 1. Схема поиска оптимального расписания

**Алгоритм ВФ.**

$TR(k) := 0, WR(k) := -\infty, PR(k) := 0, R(k) := ()$ ,  
 $k = 1, \dots, m + 1$ .

**FOR**  $l = n - m + 1, n - m + 2, \dots, n$ :

**FOR**  $k = m, m - 1, \dots, 1$ :

$W := ((k - 1)(PR(k - 1) + p_{l-k}) -$   
 $(\sum p_j - PR(k - 1) - d_{l-k}) - TR(k - 1));$

**IF**  $W > WR(k)$  **THEN**

$WR(k) := W;$

$TR(k) := (\sum p_j - PR(k - 1) - d_{l-k}) + TR(k - 1);$

$R(k) := ((l - k), R(k - 1));$

$PR(k) := PR(k - 1) + p_{l-k}.$

**END IF**

**END FOR**

**END FOR**

Оптимальное расписание – расписание при котором в порядке  $SPT$  обслуживаются требования множества  $R(1)$  в конце расписания.

Опишем процедуру выбора запаздывающих требований (см. рис. 1). Последовательно рассматриваются  $n - m + 1$  деревьев.

На *первом шаге* рассматривается дерево (цепочка) состоящая из требований  $m, m - 1, m - 2, \dots, 1$ . То есть первое запаздывающее требование  $m$ , последнее запаздывающее требование 1. На каждом уровне  $m, \dots, 1$  вычисляется "вес" поддерева (суммарное запаздывание). На каждом из последующих шагов (в каждом последующем дереве) дополнительно рассматриваются  $m$  узлов. Допустим в предыдущем дереве на уровне  $i$  выбрано требование  $j_1$ . Пусть

в текущем дереве на уровне  $i$  "добавлено" требование  $j_2$ . Среди двух веток ("начинающихся" с  $j_1$  и  $j_2$ ) выбираем ветку с наименьшим "весом". В результате сравнения выбирается одна из двух веток с наименьшим "весом"  $F(\pi') - (i - 1)p_j - d_j$ ,  $j = j_1, j_2$ , где величина  $F(\pi')$  - запаздывание требований, выбранных на соответствующей подветке. Таким образом, мы минимизируем запаздывание требований, выбранных на соответствующей подветке и  $i - 1$  требований, предшествующих требованию  $j$ .

Всего рассматривается  $n - m + 1$  деревьев. В каждом дереве добавляется  $m$  узлов. Поэтому трудоемкость алгоритма **ВФ** не превышает  $O(n^2)$  операций.

### 3.3. Алгоритм В-1 модифицированный

Кусочно-линейную функцию  $F(t)$  можно представить в виде:  $F(t) = \alpha_i t + b_i, t \in [t_i, t_{i+1}]$ ,  $i = 1, 2, \dots, m$ , где  $t_i$  – точки "излома" функции, а  $\alpha_i$  – количество запаздывающих требований при оптимальном частичном расписании на интервале  $[t_i, t_{i+1}]$ .

**Алгоритм В-1 модифицированный.**

$F(t) := 0, \pi(t) := \emptyset$ .

**FOR**  $l = n, n - 1, n - 2, \dots, 1$ :

$$F^1(t) := -(n - l + 1)t + b_1 + p_l - (d_l - d_n), \\ t \in (-\infty, p_l - (d_l - d_n)],$$

$$F^1(t) := (\alpha_i - 1)t + b_i + p_l - (d_l - d_n), t \in [t_i + p_l, t_{i+1} + p_l], \\ i = 1, 2, \dots, m,$$

$$F^2(t) := (\alpha_i - 1)t + b_i + \sum_{i=l}^n p_i - (d_l - d_n), t \in [t_i, t_{i+1}], \\ i = 1, 2, \dots, m,$$

$$F^2(t) := 0, t \in [\sum_{i=l}^n p_i - (d_l - d_n), +\infty),$$

$$\pi^1(t) = (l, \pi(t)), \pi^2(t) = (\pi(t), l).$$

**END FOR.**

В результате последовательных вычислений будет построена функция  $F(t) := \min\{F^1(t), F^2(t)\}$  и соответствующее расписание  $\pi(t)$  из  $\pi^1(t)$  и  $\pi^2(t)$  для любой точки  $t$ .

**Лемма 7.** Для случая (2) с помощью **алгоритма В-1 модифицированный** может быть построено оптимальное расписание.

**Доказательство.** При работе **алгоритма В-1** изменение частичного оптимального расписания происходит не в каждой точке  $t$ . В **алгоритме В-1**

**модифицированный** точки, где частичное оптимальное расписание не изменяется и не изменяется количество запаздывающих требований, не рассматриваются. То есть **алгоритм В-1 модифицированный** не проделывает "холостые" ходы. Поэтому алгоритмы эквивалентны с точки зрения процедуры нахождения решения.  $\square$

**Алгоритм В-1 модифицированный** перебирает не все целочисленные точки  $t$  из интервала  $[0, \sum_{j=1}^n p_j]$ , а лишь точки "излома" графика функции  $F_i(t)$ . Трудоемкость алгоритма полиномиально зависит от числа точек "излома".

### 3.4. Случай В-1 general и алгоритм его решения

Рассматривается следующий случай:

$$\begin{cases} d_1 \leq d_2 \leq \dots \leq d_n, \\ d_n - d_1 \leq p_{min}, \end{cases} \quad (4)$$

где  $p_{min} = \min_{j \in N} p_j$ .

**Лемма 8.** Пусть выполняются условия (4) и при расписании  $\pi = (j_1, j_2, \dots, j_k, \dots, j_n)$  требование  $j_k$  не запаздывает. Тогда при расписании  $\pi$  не запаздывают требования  $j_1, j_2, \dots, j_{k-1}$ .

**Доказательство.** Пусть требование  $j_k$  не запаздывает при расписании  $\pi$ . Тогда  $c_{j_k}(\pi) \leq d_{j_k}$ . Следовательно, для любого требования  $i \in \{j_1, \dots, j_{k-1}\}$  выполняется:

$$c_i(\pi) \leq c_{j_k}(\pi) - p_i \leq d_{j_k} - p_i \leq d_{j_k} - (d_{j_k} - d_i) = d_i,$$

т.е. требования  $j_1, j_2, \dots, j_{k-1}$  при расписании  $\pi$  также не запаздывают.  $\square$

**Лемма 9.** Пусть выполняются условия (4) и при расписании  $\pi = (j_1, j_2, \dots, j_k, \dots, j_n)$  требование  $j_k$  запаздывает. Тогда при расписании  $\pi$  запаздывают требования  $j_{k+1}, \dots, j_n$ .

**Доказательство.** Пусть требование  $j_k$  запаздывает при расписании  $\pi$ , т.е. выполняется  $c_{j_k}(\pi) > d_{j_k}$ . Покажем, что  $c_{j_{k+1}}(\pi) > d_{j_{k+1}}$ :

$$c_{j_{k+1}}(\pi) = c_{j_k}(\pi) + p_{j_{k+1}} \geq c_{j_k}(\pi) + p_n > d_{j_k} + p_n \geq d_{j_{k+1}}.$$

Аналогично доказывается для требований  $j_{k+2}, \dots, j_n$ .  $\square$

Несложно показать, что в случае (4) запаздывающие требования при оптимальном расписании обслуживаются в SPT порядке, кроме, быть может, первого запаздывающего требования.

**Лемма 10.** Для случая (4) существует оптимальное расписание вида  $(\pi_{EDD}, \pi_{SPT})$ ,  $\pi_{EDD}$  и  $\pi_{SPT}$  – частичные расписания, построенные по правилам EDD и SPT, соответственно.

**Доказательство.** Рассмотрим некоторое оптимальное расписание  $\pi = (\pi^1, l, \pi^2)$ , где  $l$  – первое по порядку запаздывающее требование.

Покажем, что  $d_n \geq c_l(\pi) - p_l$ . Предположим противное, пусть  $d_n < c_l(\pi) - p_l$ . Рассмотрим  $c_{j_m}(\pi)$ , где  $j_m$  – требование, обслуживаемое последним при частичном расписании  $\pi^1$ . То  $c_{j_m}(\pi) = c_l(\pi) - p_l > d_n \geq d_{j_m}$ , т.е.  $j_m$  – так же запаздывающее требование. Получили противоречие.

Рассмотрим перестановку  $\pi' = (j_1, \dots, j_{m-1}, j_m)$  из требований, которые обслуживаются при частичном расписании  $\pi^1$ .

Допустим, при расписании  $\pi^{new} = (\pi', l, \pi^2)$ ,  $l$  – не первое запаздывающее требование. Покажем, что из  $\pi'$  может запаздывать лишь требование  $j_m$

$$d_n \geq c_l(\pi) - p_l = c_l(\pi^{new}) - p_l = c_{j_m}(\pi^{new}).$$

Имеет место

$$c_{j_{m-1}}(\pi^{new}) = c_{j_m}(\pi^{new}) - p_{j_m} \leq d_n - p_{j_m} \leq d_n - (d_n - d_1) = d_1,$$

то  $c_{j_{m-1}}(\pi^{new}) \leq d_{j_{m-1}}$ . Следовательно, требование  $j_{m-1}$  не запаздывает. Аналогично можно показать, что не запаздывают требования  $j_1, j_2, \dots, j_{m-2}$ .

Учитывая, что  $c(\pi^1)$  (момент завершения обслуживания частичного расписания  $\pi^1$ ) неизменно, необходимо будет выбрать запаздывающим то требованием  $i \in \{j_1, \dots, j_m\}$ , при котором  $c(\pi^1) - d_i$  минимально, т.е. с наибольшим директивным сроком. Тогда частичное расписание  $\pi_{EDD}$

будет оптимально для требований множества  $\{\pi^1\}$ . Более того, учитывая что в рассматриваемом оптимальном расписании  $\pi = (\pi^1, l, \pi^2)$ ,  $l$  – первое запаздывающее требование, то все требования из частичного расписания  $\pi_{EDD}$  будут не запаздывающими при расписании  $(\pi_{EDD}, l, \pi^2)$ .

Для случая (4) запаздывающие требования при оптимальном расписании обслуживаются в SPT порядке, кроме, быть может, первого запаздывающего требования, то существует оптимальное расписание вида  $\pi^* = (\pi_{EDD}, l, \pi_{SPT})$ .

На основе расписания  $\pi$  построим оптимальное расписание  $\pi^* = (j_1, \dots, j_{k-1}, j_k, j_{k+1}, \dots, j_n)$ , где  $j_k$  – первое запаздывающее требование. Если  $p_{j_k} \leq p_{j_{k+1}}$ , то расписание имеет вид  $(\pi_{EDD}, \pi_{SPT})$  и лемма формально верна. Если  $d_{j_k} \geq d_{j_{k-1}}$ , то расписание имеет вид  $(\pi_{EDD}, \pi_{SPT})$  и лемма верна.

Предположим, что  $p_{j_k} > p_{j_{k+1}}$  и  $d_{j_k} < d_{j_{k-1}}$ . Если

$$c_{j_k}(\pi^*) > d_{j_{k+1}},$$

тогда

$$c_{j_{k+1}}(\pi^*) - d_{j_{k+1}} > p_{j_{k+1}}.$$

То для расписания  $\pi^{**} = (j_1, \dots, j_{k-1}, j_{k+1}, j_k, \dots, j_n)$  выполняется

$$F(\pi^{**}) = F(\pi^*) + p_{j_{k+1}} - \min\{(c_{j_{k+1}}(\pi^*) - d_{j_{k+1}}), p_{j_k}\} < F(\pi^*),$$

то расписание  $\pi^*$  не оптимально. Следовательно верно  $c_{j_k}(\pi^*) \leq d_{j_{k+1}}$ .

Рассмотрим расписание

$$\pi^{***} = (j_1, \dots, j_k, j_{k-1}, j_{k+1}, \dots, j_n),$$

$$c_{j_k}(\pi^{***}) = c_{j_k}(\pi^*) - p_{j_{k-1}} \leq d_{j_{k+1}} - (d_{j_{k+1}} - d_{j_k}) = d_{j_k}.$$

Тогда

$$F(\pi^{***}) = F(\pi^*) + (c_{j_k}(\pi^*) - d_{j_{k-1}}) - (c_{j_k}(\pi^*) - d_{j_k}) \leq F(\pi^*),$$

то есть существует оптимальное расписание  $\pi^{***}$  вида  $(\pi_{EDD}, \pi_{SPT})$ .

Следовательно, существует оптимальное расписание, при котором  $p_{j_k} \leq p_{j_{k+1}}$  или  $d_{j_k} \geq d_{j_{k-1}}$ , т.е. расписание имеет вид  $(\pi_{EDD}, \pi_{SPT})$ . Лемма доказана.  $\square$

**Лемма 11.** Для случая (4) существует оптимальное расписание вида  $(\pi_{LPT}, l, \pi_{SPT})$ ,  $\pi_{LPT}$  и  $\pi_{SPT}$  – частичные расписания, построенные по правилам LPT и SPT, соответственно.

**Доказательство.** Несложно показать, что для каждого оптимального расписания  $\pi = (\pi_{EDD}, \pi_{SPT})$  существует расписание  $\pi' = (\pi_{LPT}, l, \pi_{SPT})$ , где  $\{\pi_{EDD}\} = \{\pi_{LPT}\} \cup \{l\}$ . Причем  $F(\pi) = F(\pi')$ .  $\square$

**Алгоритм В-1 general** решения исходного примера  $\langle \{p_j, d_j\}_{j \in N}, t_0 \rangle$  в случае (4). Входными параметрами алгоритма являются  $p_j, d_j, j \in N$ , и  $t_0$ .

0.  $\pi_n(t) := (n), F_n(t) := \max\{0, p_n + t_0 - t\};$

1. **FOR**  $l = n - 1, n - 2, \dots, 1:$

$\pi^1 := (l, \pi_{l+1}(t - p_l)),$

Если существует представление  $\pi_{l+1}(t) = (\pi_\alpha, \pi_\beta),$   
что выполняется

$p_l \geq p_i, \forall i \in \{\pi_\alpha\}$  и  $p_l \leq p_i, T_i(\pi_{l+1}(t)) > 0,$   
 $\forall i \in \{\pi_\beta\},$

то  $\pi^2 := (\pi_\alpha, l, \pi_\beta),$

иначе  $\pi^2 := \pi^1;$

$F(\pi^1) := \max\{0, p_l - d_l(t)\} + F_{l+1}(t - p_l);$

$F_l(t) := \min\{F(\pi^1), F(\pi^2)\};$

$\pi_l(t) := \arg \min\{F(\pi^1), F(\pi^2)\}.$

2. Алгоритм возвращает расписание  $\pi_1(d_n)$  и соответствующее ему значение суммарного запаздывания  $F_1(d_n).$

Шаги основного цикла (1.) выполняются для всех целых  $t \in [0, \sum p_j].$

**Теорема 3.** Алгоритм В-1 general на шаге  $l$  для множества требований  $N_l = \{l, l + 1, \dots, n\}$  в каждой точке  $t \in [0, \sum p_j]$  строит оптимальное частичное расписание  $\pi_l(t).$

**Доказательство.** Воспользуемся методом математической индукции. Очевидно, что на шаге  $n$  алгоритмом в каждой точке  $t$  построено оптимальное расписание  $\pi_n(t) = (n).$

Предположим, что алгоритмом на шаге  $l + 1$  в каждой точке  $t$  для множества требований  $N_{l+1} = \{l + 1, \dots, n\}$  построено оптимальное расписание  $\pi_{l+1}(t)$ .

Покажем, что на шаге  $l$  для множества требований  $N_l = \{l, l+1, \dots, n\}$  в точке  $t$  одно из расписаний  $\pi^1 = (l, \pi_{l+1}(t - p_l))$  или  $\pi^2 = (\pi_\alpha, l, \pi_\beta)$  оптимально.

1. Если на шаге  $l$  для множества требований  $N_l = \{l, l + 1, \dots, n\}$  в точке  $t$  существует оптимальное частичное расписание  $\pi'$ , при котором требование  $l$  не запаздывает, тогда частичное расписание  $\pi^1 = (l, \pi_{l+1}(t - p_l))$  оптимально.

Предположим обратное. Пусть при расписании  $\pi' = (j_1, \dots, j_m, l, j_{m+1}, \dots, j_{n-l+1})$  требование  $l$  не запаздывает и выполняется  $F(\pi^1) > F(\pi')$ .

Рассмотрим расписание

$$\pi'' = (l, j_1, \dots, j_m, j_{m+1}, \dots, j_{n-l+1}).$$

Очевидно, что при этом расписании требование  $j_m$  не запаздывает, так как  $d_l \leq d_{j_m}$ .

Выполняется  $F(\pi') = F(\pi'') < F(\pi^1)$ . Но в этом случае в точке  $t - p_l$  на шаге  $l + 1$  построено неоптимальное расписание  $\pi_{l+1}(t - p_l)$ , так как при частичном расписании  $(j_1, \dots, j_m, j_{m+1}, \dots, j_{n-l+1})$  значение целевой функции в точке  $(t - p_l)$  меньше. Получили противоречие. Значит частичное расписание  $\pi^1 := (l, \pi_{l+1}(t - p_l))$  оптимально.

2. Рассмотрим расписание  $\pi^2 = (\pi_\alpha, l, \pi_\beta)$ . Определим расписание  $\pi' = (\pi'_\alpha, l, \pi'_\beta)$ , где  $|\{\pi_\beta\}| = |\{\pi'_\beta\}|$ .

Покажем, что  $F(\pi^2) \leq F(\pi')$  или  $F(\pi^1) \leq F(\pi')$ .

Если  $T_l(\pi^2) = 0$ , то очевидно,  $F(\pi^1) \leq F(\pi')$  (см. п.1).

Рассмотрим случай  $T_l(\pi^2) > 0$ .

Обозначим  $|\{\pi_\beta\}| = |\{\pi'_\beta\}| = s$ . Пусть  $\pi_\beta = (j_1, \dots, j_s)$ ,  $\pi'_\beta = (j'_1, \dots, j'_s)$ .

- а) Если при расписании  $(\pi'_\alpha, \pi'_\beta)$  требование  $j'_1$  запаздывает. Предположим для расписаний  $\pi^2, \pi'$  имеет место  $F(\pi^2) > F(\pi')$ , тогда

$$F(\pi'_\alpha, \pi'_\beta) = F(\pi'_\alpha, l, \pi'_\beta) - p_l s.$$

В силу выбора  $\pi_\beta$  также выполняется

$$F(\pi_\alpha, \pi_\beta) = F(\pi_\alpha, l, \pi_\beta) - p_l s.$$

Тогда

$$F(\pi'_\alpha, \pi'_\beta) < F(\pi_\alpha, \pi_\beta),$$

что невозможно, так как  $(\pi_\alpha, \pi_\beta)$  – оптимальное частичное расписание на шаге  $l + 1$  в точке  $t$ .

Следовательно,  $F(\pi^2) \leq F(\pi')$ .

- б) Если при расписании  $(\pi'_\alpha, \pi'_\beta)$  требование  $j'_1$  не запаздывает. Тогда  $P(\{\pi'_\alpha, j'_1\}) \leq d_{j'_1}(t)$ .

Предположим, имеет место  $F(\pi^2) > F(\pi')$ .

Выполняется  $\forall i \in \{\pi'_\alpha\}, T_i(\pi'_\alpha) = 0$ .

Пусть  $j_k \in \pi_\beta \cap \pi'_\alpha$  и  $\pi' = (\pi'_{\alpha_1}, j_k, \pi'_{\alpha_2}, l, \pi'_\beta)$ .

Для расписания  $\pi'' = (\pi'_{\alpha_1}, l, \pi'_{\alpha_2}, j_k, \pi'_\beta)$  выполняется  $F(\pi') \geq F(\pi'')$ , так как  $p_{j_k} > p_l, d_{j_k} > d_l$ .

$$c_l(\pi'') \leq P(\{\pi'_\alpha\}) \leq d_{j'_1}(t) - p_{j'_1} \leq d_l(t).$$

При расписании  $\pi''$  требование  $l$  не запаздывает. Следовательно, выполняется условие п.1 и поэтому  $F(\pi^1) \leq F(\pi')$ .

Таким образом, на шаге  $l$  в точке  $t$  одно из расписаний  $\pi^1$  или  $\pi^2$  является оптимальным.  $\square$

**Лемма 12.** Трудоемкость алгоритма **В-1 general**  $O(n^2 \sum p_j)$ .

**Доказательство.** Шаг алгоритма для фиксированных  $l$  и  $t$  выполняется за  $O(n)$  операций. Основной шаг алгоритма для фиксированного требования  $l$  выполняется за  $O(\sum p_j)$  операций, так как  $p_j \in \mathbb{Z}^+$ ,  $\forall j \in N$ , и достаточно рассматривать только целочисленные точки  $t$  на указанном интервале. Следовательно, трудоемкость всего алгоритма составляет  $O(n^2 \sum p_j)$  операций.  $\square$

### 3.5. Случай, когда $k = 1$ и $d_n - d_1 \leq 1$

В данном параграфе предлагается полиномиальный алгоритм решения трудоемкости  $O(n^2)$ . Предполагается, что параметры требований удовлетворяют условиям

$$\begin{cases} d_1 \leq d_2 \leq \dots \leq d_n, \\ d_n - d_1 \leq 1. \end{cases} \quad (5)$$

Приведем алгоритм решения примеров в случае (5). Будем обозначать  $z = \lfloor d_n \rfloor$ ,  $z$  – целая часть величины  $d_n$ .

**Алгоритм С-1** решения исходного примера в случае (5).

0.  $S := t_0 + \sum_{j=1}^n p_j$ ,  $N' := N$ ,  $\pi^* := \emptyset$ ,  $\pi_D := (1, 2, \dots, n)$ ;
1. **WHILE**  $\{j \in N' : S - p_j \leq z\} = \emptyset$  **AND**  $N' \neq \emptyset$  **DO**:  
 $\pi^* := (j^*, \pi^*)$ , где  $j^* := \arg \max_{j \in N'} \{d_j : p_j = \max_{i \in N'} p_i\}$ ;  
 $S := S - p_{j^*}$ ,  $N' := N' \setminus \{j^*\}$ ,  $\pi_D := \pi_D \setminus \{j^*\}$ ;
2. **IF**  $|N'| = 1$ , т.е.  $N' = \{j\}$ , **THEN**  $\pi^* := (j, \pi^*)$ ; **STOP**.
3. **FOR ALL**  $j \in N' : S - p_j \leq z + 1$  **DO**:  
**FOR ALL**  $i \in N' \setminus \{j\}$  **DO**:  
 $\pi_{ij} := (\pi_D \setminus \{i, j\}, i, j)$ ;
4.  $\pi^* := (\pi, \pi^*)$ , где  $\pi := \arg \min_{i,j} F(\pi_{ij})$ .

Оптимальность расписания  $\pi^*$ , построенного алгоритмом **С-1**, в рассматриваемом случае (5) устанавливается теоремой 4. Доказательство теоремы 4

основывается на леммах 13–15. Леммой 13 обосновывается шаг 1 алгоритма.

Будем обозначать через  $\Pi^*(N, t)$  множество оптимальных расписаний обслуживания требований множества  $N$  с момента времени  $t$ .

**Лемма 13.** Пусть параметры требований множества  $N' \subseteq N$  удовлетворяют (5) и  $S := t_0 + \sum_{j \in N'} p_j$ . Если  $S - p_j > z = \lfloor d_n \rfloor$  для всех  $j \in N'$ , то существует оптимальное расписание  $\pi^* \in \Pi^*(N', t_0)$  такое, что  $\pi^* = (\tilde{\pi}, j^*)$ , где требование  $j^* = \arg \max_{j \in N'} \{d_j : p_j = \max_{i \in N'} p_i\}$  и расписание  $\tilde{\pi} \in \Pi^*(N' \setminus \{j^*\}, t_0)$ .

**Доказательство.** Предположим, что для всех оптимальных расписаний выполняется  $\pi = (\pi_1, j^*, \pi_2, \alpha, j)$ . Построим расписание  $\pi' = (\pi_1, j, \pi_2, \alpha, j^*)$ ,  $\alpha, j \in N'$ . Заметим, что  $p_{j^*} + d_{j^*} = \max_{j \in N'} \{p_j + d_j\}$ . Это следует из того факта, что  $p_j \in \mathbb{Z}^+$  и  $d_i - d_j \leq 1$  для всех  $i, j \in N'$ .

Если  $p_{j^*} = p_j$ , то  $d_{j^*} \geq d_j$  и выполняется:

- (а)  $F(\pi') - F(\pi) = 0$ , если  $c_{j^*}(\pi) \geq d_j$ ;
- (б)  $F(\pi') - F(\pi) \leq d_j - d_{j^*} \leq 0$ , если  $c_{j^*}(\pi) < d_j$ .

Рассмотрим случай, когда  $p_{j^*} > p_j$ .

**1)** Пусть  $\{(\pi_2, \alpha)\} = \emptyset$ , т.е.  $\pi = (\pi_1, j^*, j)$ . Так как  $S - p_i > z$  для всех  $i \in N'$ , то имеем  $c_{j^*}(\pi) > z + 1$  и  $c_j(\pi') \geq z + 1$ . Следовательно,  $F(\pi') - F(\pi) \leq p_j - p_{j^*} < 0$ .

**2)** Пусть  $\{(\pi_2, \alpha)\} \neq \emptyset$ . В этом случае имеем:

$$\begin{aligned} T_j(\pi) &= S - d_j, & T_j(\pi') &= \max\{0, c_{j^*}(\pi) - p_{j^*} + p_j - d_j\}, \\ T_\alpha(\pi) &= S - p_j - d_\alpha, & T_\alpha(\pi') &= S - p_{j^*} - d_\alpha, \\ T_{j^*}(\pi) &= \max\{0, c_{j^*}(\pi) - d_{j^*}\}, & T_{j^*}(\pi') &= S - d_{j^*}. \end{aligned}$$

Рассмотрим следующие подслучаи:

**а)** Если  $c_{j^*}(\pi) \leq d_{j^*}$ , тогда  $c_j(\pi') \leq d_j$  и выполняется  $F(\pi') - F(\pi) \leq (p_j + d_j) - (p_{j^*} + d_{j^*}) \leq 0$ .

**б)** Если  $c_{j^*}(\pi) > d_{j^*}$  и  $c_j(\pi') \leq d_j$ , тогда будем иметь  $F(\pi') - F(\pi) \leq (d_j - c_{j^*}(\pi)) - (p_{j^*} - p_j)$ . Поскольку  $c_{j^*}(\pi) > d_{j^*}$ , то выполняется  $d_j - c_{j^*}(\pi) < d_j - d_{j^*} \leq 1$ . И, наконец, из  $p_{j^*} > p_j$  и  $p_i \in \mathbb{Z}^+$ ,  $i \in N'$ , имеем  $p_{j^*} - p_j \geq 1$  и  $F(\pi') - F(\pi) \leq 0$ .

**в)**  $c_{j^*}(\pi) > d_{j^*}$  и  $c_j(\pi') > d_j$ . В этом случае выполняется  $F(\pi') - F(\pi) \leq 2(p_j - p_{j^*}) < 0$ .

Следовательно,  $F(\pi') \leq F(\pi)$ . Так как расписание  $\pi$  оптимально, то и расписание  $\pi'$  также оптимально, причем на последнем месте обслуживается требование  $j^*$ . Лемма доказана.  $\square$

В леммах 14 и 15 показывается оптимальность шага 3 алгоритма **С-1**. Лемма 14 устанавливает, в каком порядке должно строиться оптимальное расписание  $\pi^*$ , если множество  $\{j \in N' : S - p_j \leq z\} \neq \emptyset$ . Лемма 15 показывает, что если  $S \leq z$ , то все неупорядоченные требования должны быть включены в начало расписания  $\pi^*$  в порядке убывания директивных сроков.

**Лемма 14.** Пусть параметры требований множества  $N' \subseteq N$  удовлетворяют условиям (5) и  $S := t_0 + \sum_{j \in N'} p_j$ ,  $\bar{N} := \{i \in N' : S - p_i \leq z\}$ . Если  $\bar{N} \neq \emptyset$ , тогда существует оптимальное расписание  $\pi^* \in \Pi^*(N', t_0)$  такое, что выполняется  $\pi^* = (\tilde{\pi}, \mu)$ , где  $c_\mu(\pi^*) - p_\mu \leq z + 1$  и  $\tilde{\pi} \in \Pi^*(N' \setminus \{\mu\}, t_0)$ .

**Доказательство.** Заметим, что  $c_\mu(\pi^*) = S$ . Предположим, что для всех оптимальных расписаний выполняется  $\pi = (\pi_1, \nu, \pi_2, \alpha, j)$ , при этом  $S - p_j > z + 1$  и  $\nu$  - некоторое требование из множества  $\bar{N}$ . Построим расписание  $\pi' = (\pi_1, j, \pi_2, \alpha, \nu)$ . Так как  $j \notin \bar{N}$  и  $S - p_\nu \leq z$ , то выполняется  $p_\nu > p_j + 1$ . Поскольку продолжительности обслуживания требований целочисленны, то  $p_\nu \geq p_j + 2$ .

**1)** Рассмотрим случай, когда  $\{(\pi_2, \alpha)\} = \emptyset$ . При этом, имеем  $\pi = (\pi_1, \nu, j)$  и  $\pi' = (\pi_1, j, \nu)$ , а также

$$\begin{aligned} T_\nu(\pi) &= S - p_j - d_\nu, & T_\nu(\pi') &= S - d_\nu, \\ T_j(\pi) &= S - d_j, & T_j(\pi') &= \max\{0, S - p_\nu - d_j\}. \end{aligned}$$

Из условия  $p_\nu > p_j$  следует

$$F(\pi') - F(\pi) = \max\{0, S - p_\nu - d_j\} - S + p_j + d_j \leq 0.$$

**2)** Рассмотрим случай  $\{(\pi_2, \alpha)\} \neq \emptyset$ . В этом случае получаем

$$\begin{aligned} T_\nu(\pi) &= \max\{0, c_\nu(\pi) - d_\nu\}, & T_\nu(\pi') &= S - d_\nu, \\ T_j(\pi) &= S - d_j, & T_j(\pi') &= 0, \\ T_\alpha(\pi) &= S - p_j - d_\alpha, & T_\alpha(\pi') &= \max\{0, S - p_\nu - d_\alpha\}. \end{aligned}$$

Необходимо рассмотреть следующие подслучаи.

**а)** Пусть  $T_\nu(\pi) = 0$  и  $T_\alpha(\pi') = 0$ . В этом случае имеем  $F(\pi') - F(\pi) \leq p_j + d_j - S + d_\alpha - d_\nu$ . Из условий  $d_\alpha - d_\nu \leq 1$ ,  $S - p_j > z + 1$  и  $p_j \in \mathbb{Z}^+$  следует, что  $S - p_j \geq z + 2 \geq d_j + 1$  и  $p_j + d_j - S \leq -1$ . Следовательно,  $F(\pi') - F(\pi) \leq 0$ .

**б)** Пусть  $T_\nu(\pi) > 0$  и  $T_\alpha(\pi') = 0$ . В этом случае имеем  $F(\pi') - F(\pi) \leq p_j + d_j - S + d_\alpha - c_\nu(\pi)$ . Согласно условию  $c_\nu(\pi) > d_\nu$  получаем  $d_\alpha - c_\nu(\pi) \leq d_\alpha - d_\nu \leq 1$ . Следовательно,  $F(\pi') - F(\pi) \leq 0$ .

**в)** Пусть  $T_\nu(\pi) = 0$  и  $T_\alpha(\pi') > 0$ . В этом случае получаем  $F(\pi') - F(\pi) \leq (p_j + d_j) - (p_\nu + d_\nu)$ . Из условий  $p_j - p_\nu \leq -2$  и  $d_j - d_\nu \leq 1$  следует, что  $F(\pi') - F(\pi) < 0$ .

**г)** Пусть  $T_\nu(\pi) > 0$  и  $T_\alpha(\pi') > 0$ . Поскольку  $c_\nu(\pi) \leq d_\nu$ , то выполняется  $F(\pi') - F(\pi) \leq (p_j + d_j) - (c_\nu(\pi) + d_\nu) \leq (p_j + d_j) - (p_\nu + d_\nu) \leq 0$ .

Следовательно, если  $\bar{N} \neq \emptyset$ , тогда  $F(\pi') - F(\pi) \leq 0$ , и каждое расписание  $\pi = (\pi_1, \nu, \pi_2, \alpha, j)$ , при котором разность  $S - p_j > z + 1$ , без потери оптимальности может быть сведено к расписанию  $\pi'$ , при котором требование  $\nu$  обслуживается последним по порядку. Так как требование  $\nu \in \bar{N}$ , то будем иметь  $S - p_\nu \leq z < z + 1$ . Лемма доказана.  $\square$

**Лемма 15.** Пусть параметры требований множества  $N' \subseteq N$ ,  $|N'| = m$ , удовлетворяют условиям (5) и величина  $S := t_0 + \sum_{j \in N'} p_j$ . Если  $S \leq z = \lfloor d_n \rfloor$ , тогда существует оптимальное расписание  $\pi^* \in \Pi^*(N', t_0)$   $\pi^* = (j_1, j_2, \dots, j_m)$ , при котором  $d_{j_1} \leq d_{j_2} \leq \dots \leq d_{j_m}$ .

**Доказательство.** Предположим, что для всех оптимальных расписаний выполняется  $\pi = (\pi_1, j_m, \pi_2, j)$ , где  $d_j < d_{j_m}$ . Построим расписание  $\pi' = (\pi_1, j, \pi_2, j_m)$ . Заметим, что при расписаниях  $\pi$  и  $\pi'$  запаздывает не более одного требования ( $j$  и  $j_m$ , соответственно). Это следует из того, что  $p_j \in \mathbb{Z}^+$ ,  $|d_j - z| \leq 1$  и  $S \leq z$ .

Получаем  $F(\pi') - F(\pi) \leq \max\{0, S - d_{j_m}\} - \max\{0, S - d_j\}$ . Поскольку  $d_j < d_{j_m}$ , то выполняется  $F(\pi') - F(\pi) \leq 0$ .

Следовательно, существует оптимальное расписание, при котором требование с наибольшим директивным сроком  $d_{j_m}$  обслуживается после всех требований множества  $N' \setminus \{j_m\}$ . Требования множества  $N' \setminus \{j_m\}$  при этом не запаздывают и могут быть обслужены в произвольном порядке, в частности, по неубыванию директивных сроков. Лемма доказана.  $\square$

**Теорема 4.** Расписание  $\pi^*$ , построенное с помощью алгоритма С-1, является оптимальным для случая (5) и может быть найдено за  $O(n^2)$  операций.

**Доказательство.** На шаге 1 алгоритма С-1 рассматривается ситуация  $\{i \in N' : S - p_i \leq z\} = \emptyset$ , и расписание  $\pi^*$  строится в соответствии с леммой 13.

На шаге 3 алгоритма из расписания  $\pi_D$  строятся расписания  $\pi_{ij} = (\pi_D \setminus \{i, j\}, i, j)$  для всех  $j \in N', i \in N' \setminus \{j\}$ , для которых  $S - p_j \leq z + 1$ . В этом случае имеем  $S - p_j \leq z + 1$  и  $S - p_j - p_i \leq z$ , и согласно лемме 15, все неупорядоченные требования могут быть обслужены в порядке неубывания директивных сроков. Поскольку в ходе работы алгоритма расписания  $\pi_{ij}$  строятся для всех возможных пар  $i$  и  $j$ , то расписание  $\pi^*$  является оптимальным для случая (5).

Вычислим трудоемкость алгоритма. Шаг 0 алгоритма выполняется за  $O(n)$  операций. Шаг 1 алгоритма выполняется за  $O(n^2)$  операций. На третьем шаге строятся не более чем  $O(n^2)$  расписаний. На четвертом шаге среди построенных расписаний  $\pi_{ij}$  выбирается наилучшее. При каждом расписании  $\pi_{ij}$  запаздывает не более трех последних по порядку требований. Поэтому, чтобы вычислить значение величины  $F(\pi_{ij})$  для каждого расписания  $\pi_{ij}$ , необходимо выполнить  $O(1)$  операций. Значит, шаг 4 алгоритма выполняется за  $O(n^2)$  операций. Следовательно, **алгоритм С-1** выполняется за  $O(n^2)$  операций. Теорема доказана.  $\square$

### 3.6. Алгоритм решения в случае $1 < k < n$

В случае  $1 < k < n$  параметры требований удовлетворяют условиям

$$\begin{cases} d_1 \leq d_2 \leq \dots \leq d_n, \\ p_1 \geq p_2 \geq \dots \geq p_n, \\ d_{\beta_1} - d_{\alpha_1} \leq p_{\beta_1}, & \alpha_1 = 1, \\ d_{\beta_2} - d_{\alpha_2} \leq p_{\beta_2}, & \alpha_2 = \beta_1 + 1, \\ \dots \\ d_{\beta_k} - d_{\alpha_k} \leq p_{\beta_k}, & \beta_k = n. \end{cases} \quad (6)$$

Предлагается псевдо-полиномиальный алгоритм решения примеров проблемы в случае (6). В данном параграфе будем использовать обозначения:  $\Sigma_i(a) = \sum_{j=1}^i a_j$  для некоторого набора величин  $(a_1, a_2, \dots, a_i)$ .

**Лемма 16.** Если выполняются ограничения (1), то существует оптимальное расписание  $\pi^* \in \Pi^*(N, t_0)$ , при котором для всех  $l = n - 1, n - 2, \dots, 1$  выполняется:

- или  $(l \rightarrow j)_{\pi^*}$  для всех  $j \in \{l + 1, l + 2, \dots, n\}$ ;
- или  $((l + 1) \rightarrow l)_{\pi^*}$ .

**Доказательство.** Очевидно, что для требований  $n, n - 1$  будем иметь или  $(n \rightarrow (n - 1))_{\pi^*}$ , или  $((n - 1) \rightarrow n)_{\pi^*}$ . Поэтому, рассмотрим случай  $l \leq n - 2$ . Предположим, что для всех оптимальных расписаний выполняется

$\pi = (\pi_1, j, \pi_2, l, \pi_3, l + 1, \pi_4)$ , для некоторого  $l \in \{1, 2, \dots, n - 2\}$  и для некоторого  $j \in \{l + 2, \dots, n\}$ . Построим расписания  $\pi' = (\pi_1, \pi_2, l, j, \pi_3, l + 1, \pi_4)$  и  $\pi'' = (\pi_1, j, \pi_2, l + 1, \pi_3, l, \pi_4)$ . Для требований  $j, l, l + 1$  выполняется  $p_l \geq p_{l+1} \geq p_j$  и  $d_l \leq d_{l+1} \leq d_j$ .

Рассмотрим следующие случаи.

1) Пусть  $c_l(\pi) \leq d_l$ . В этом случае оба требования  $j$  и  $l$  не запаздывают при расписаниях  $\pi$  и  $\pi'$ , поэтому выполняется  $F(\pi') \leq F(\pi)$ .

2) Пусть  $c_l(\pi) > d_l$  и  $c_l(\pi) \leq d_{l+1}$ . Так как  $d_{l+1} \leq d_j$ , то выполняется  $c_j(\pi') = c_l(\pi) \leq d_{l+1} \leq d_j$ . Получаем

$$F(\pi') - F(\pi) \leq \max\{0, c_l(\pi) - p_j - d_l\} - (c_l(\pi) - d_l) \leq 0.$$

3) Пусть  $c_l(\pi) > d_l$  и  $c_l(\pi) > d_{l+1}$ . Так как  $p_l \geq p_{l+1}$ , то выполняется  $c_l(\pi'') = c_{l+1}(\pi)$  и  $c_{l+1}(\pi'') \leq c_l(\pi)$ . В этом случае получаем

$$F(\pi'') - F(\pi) \leq \max\{0, c_{l+1}(\pi'') - d_{l+1}\} + d_{l+1} - c_l(\pi) \leq 0.$$

Следовательно, расписание  $\pi = (\pi_1, j, \pi_2, l, \pi_3, l + 1, \pi_4)$  без потери оптимальности может быть преобразовано к расписанию  $\pi'$  или  $\pi''$ . Лемма доказана.  $\square$

**Лемма 17.** Если выполняются ограничения (6), то существует оптимальное расписание  $\pi^* \in \Pi^*(N, t_0)$ , при котором для всех  $l = n, n - 1, \dots, 1$ , для всех требований  $\nu = m, m + 1, \dots, k$ , где  $l \in M_m$ , верно

- или  $(j \rightarrow l)_{\pi^*}$  для всех  $j \in M_\nu \setminus \{\alpha_m, \dots, l\}$ ,
- или  $(l \rightarrow j)_{\pi^*}$  для всех  $j \in M_\nu \setminus \{\alpha_m, \dots, l\}$ .

**Доказательство.** Предположим противное, что для всех оптимальных расписаний  $\pi \in \Pi^*(N, t_0)$  для некоторого требования  $l \in \{n, \dots, 1\}$  и для некоторого подмножества  $M_\nu$ , где  $\nu \in \{m, \dots, k\}$ ,  $l \in M_m$ , существуют такие два требования  $i, j \in M_\nu \setminus \{\alpha_m, \dots, l\}$ , что  $(i \rightarrow l)_\pi$  и  $(l \rightarrow j)_\pi$ , т.е.  $\pi = (\pi_1, i, \pi_2, l, \pi_3, j, \pi_4)$ .

В силу условий (6) выполняется  $p_i \leq p_l$ ,  $d_i \geq d_l$ , а также  $p_j \leq p_l$ ,  $d_j \geq d_l$  и  $|d_i - d_j| \leq p_{\beta_\nu}$ . Построим расписания  $\pi' = (\pi_1, \pi_2, l, i, \pi_3, j, \pi_4)$  и  $\pi'' = (\pi_1, i, \pi_2, j, \pi_3, l, \pi_4)$ .

Если  $|M_\nu \setminus \{\alpha_m, \dots, l\}| \leq 1$  (например, подмножество  $M_\nu$  состоит из одного требования), то два требования  $i$  и  $j$  мы, очевидно, выбрать не сможем и лемма формально верна.

Рассмотрим следующие случаи.

**1)** Пусть  $c_l(\pi) \leq d_l$ . В этом случае получаем, что для требований  $i$  и  $l$  верно  $c_i(\pi') = c_l(\pi) \leq d_l \leq d_i$ . Оба требования  $i$  и  $l$  не запаздывают при расписаниях  $\pi$  и  $\pi'$ , поэтому выполняется  $F(\pi') \leq F(\pi)$ .

**2)** Пусть  $c_l(\pi) > d_l$  и  $c_l(\pi) \leq d_j$ . В силу того, что  $|d_i - d_j| \leq p_{\beta_\nu} \leq p_l$  и  $c_i(\pi') = c_l(\pi)$  получаем, что требование  $i$  не запаздывает при расписании  $\pi$ , т.е.  $c_i(\pi) \leq d_i$ . Тогда выполняется

$$F(\pi') - F(\pi) \leq \max\{0, c_l(\pi) - d_i\} - \max\{0, c_l(\pi) - p_i - d_l\} - \\ - c_l(\pi) + d_l \leq 0.$$

**3)** Пусть  $c_l(\pi) > d_l$  и  $c_l(\pi) > d_j$ . В этом случае имеем  $F(\pi'') - F(\pi) \leq \max\{0, c_l(\pi) - p_l + p_j - d_j\} + c_j(\pi) - d_l - c_l(\pi) + d_l - c_j(\pi) + d_j \leq \max\{0, c_l(\pi) - p_l + p_j - d_j\} - c_l(\pi) + d_j \leq 0$ .

Таким образом, каждое оптимальное расписание  $\pi = (\pi_1, i, \pi_2, l, \pi_3, j, \pi_4)$  без потери оптимальности может быть преобразовано к расписанию  $\pi'$  или  $\pi''$ . Проведя подобные рассуждения последовательно для всех  $l = n, n-1, \dots, 1$ , мы приходим к истинности леммы.  $\square$

Необходимо отметить, что примеры  $x_1 = \langle \{p_j, d_j\}_{j \in N}, t_0 \rangle$  и  $x_2 = \langle \{p_j, d'_j\}_{j \in N}, t_0 \rangle$  эквивалентны (т.е. имеют одно и то же множество оптимальных расписаний), когда директивный срок  $d'_j = \min\{\max\{t_0 + p_j, d_j\}, t_0 + \sum_{i=1}^n p_i\}$ . Действительно, если  $d_j > t_0 + \sum_{i=1}^n p_i$ , то  $d'_j = t_0 + \sum_{i=1}^n p_i$  и требование  $j$  не запаздывает при любом расписании и может быть обслужено последним для обоих примеров. Если же верно  $d_j < t_0 + p_j$ , то  $d'_j = t_0 + p_j$  и требование  $j$  запаздывает при любом расписании

для обоих примеров и в этом случае выполняется равенство  $F^*(x_1) - F^*(x_2) = \sum_{j=1}^n t_0 + p_j - d_j = \text{const}$ .

Следовательно, будем рассматривать примеры, для которых  $d_n - d_1 \leq \sum_{j=1}^n p_j$ .

Если выполняется  $t \leq t_0 + p_n$ , то для примера  $\langle \{p_j, d_j(t)\}_{j \in N}, 0 \rangle$ , по аналогии со случаем  $k = 1$ , каждое требование будет запаздывать при любом расписании, построенном с момента времени 0 и SPT расписание в порядке неубывания продолжительностей обслуживания  $(n, n-1, \dots, 1)$  будет оптимальным.

Если  $t \geq t_0 + 2 \sum_{j=1}^n p_j$ , то учитывая неравенство  $\sum_{j=1}^n p_j \geq d_n - d_1$ , получаем следующее очевидное неравенство  $t + \sum_{j=1}^n p_j \geq t_0 + 2 \sum_{j=1}^n p_j + d_n - d_1$ . Отсюда следует

$$\sum_{j=1}^n p_j \leq d_1 - d_n + t - t_0 = d_1(t) \leq d_j(t), j \in N.$$

Следовательно, при любом расписании запаздывающим будет не более одного требования, которое обслуживается последним по порядку, поэтому EDD расписание  $(1, 2, \dots, n)$  будет оптимальным.

Определим через  $\gamma(j)$  номер подмножества, которое содержит требование  $j$ , т.е.  $j \in M_{\gamma(j)}$ , и пусть величина  $\gamma(\pi) = \min_{j \in \{\pi\}} \gamma(j)$ . Рассмотрим некоторое расписание  $\pi$ ,  $\{\pi\} = N_l$ ,  $N_l = \{l, l+1, \dots, n\}$ ,  $l = n, n-1, \dots, 1$ . Определим процедуру  $G(\pi)$ , которая производит разбиение расписания  $\pi = (j_1, \dots, j_m)$  на совокупность частичных расписаний  $(\pi_1, \pi_2, \dots, \pi_g)$ ,  $g \leq k$ . При этом

$\pi_i = \text{SubSequence}(\pi \setminus \{\pi_{i+1}\} \cup \dots \cup \{\pi_g\})$ ,  $i = 1, \dots, g$ , где

$\text{SubSequence}(\pi) := (j_\alpha, \dots, j_m)$ ,  $\alpha = \arg \min_{1 \leq i \leq m} \{\gamma(j_i) \geq \gamma(j_m)\}$ .

Если для некоторого  $\pi_i = (j_\mu, \dots, j_\nu)$  выполняется неравенство  $\gamma(\pi_i) < \gamma(j_\nu)$ , то будем полагать  $G(\pi) = \emptyset$ . Для

SPT расписания  $\pi = (n, n - 1, \dots, 1)$  имеем  $G(\pi) = (\pi)$ , и для EDD расписания  $\pi = (1, 2, \dots, n)$  имеем  $G(\pi) = (\pi_1, \dots, \pi_k)$ , где  $\{\pi_i\} = M_i$ ,  $i = 1, \dots, k$ . Заметим, что если для некоторого расписания  $\pi$  выполняется  $G(\pi) = \emptyset$ , то существует расписание  $\pi'$ , при котором  $G(\pi') \neq \emptyset$  и верно  $F(\pi') \leq F(\pi)$ . Справедливость этого следует из лемм 16 и 17.

Определим через  $G_l$ ,  $l = n, \dots, 1$ , как упорядоченное множество наборов из четырех элементов для расписания  $\pi_l$ ,  $\{\pi_l\} = N_l$ ,  $G_l = \{\langle \nu_i, f_i, P_i, \pi_i \rangle\}_{i=1, \dots, g}$ . При этом выполняется  $G(\pi_l) = (\pi_1, \pi_2, \dots, \pi_g)$ ;  $\nu_i = \gamma(\pi_i)$ ;  $P_i = \sum_{j \in \{\pi_i\}} p_j$  - продолжительность частичного расписания  $\pi_i$ ;  $f_i = F(\pi_i, t_0 + \Sigma_{i-1}(P))$  - значение суммарного запаздывания при расписании  $\pi_i$ .

Приведем алгоритм решения примеров в случае (6). Через  $\nu_i, f_i, P_i, \pi_i$  обозначаются элементы набора  $G_{l+1}(t)$  для текущих значений  $l$  и  $t$ ;  $F_{g+1} = P_{g+1} = 0$ ,  $\pi_{g+1} = G_{g+1}(t) = \emptyset$ .

**Алгоритм В-к** решения примеров в случае (6).

$$0. \pi_n(t) := (n), F_n(t) := \max\{0, p_n + t_0 - t\}, \\ G_n(t) = \{ \langle k, F_n(t), p_n, \pi_n(t) \rangle \};$$

1. **FOR**  $m = k, k - 1, \dots, 1; l = \beta_m, \beta_m - 1, \dots, \alpha_m, l < n:$

$$i = 1, 2, \dots, g + 1:$$

$$\pi^i := (\pi_1 \dots, \pi_{i-1}, l, \pi_{\alpha_{\nu_i}}(t - \sum_{i-1}(P) - pl)),$$

$$F(\pi^i) := \sum_{i-1}(f) + \max\{0, \sum_{i-1}(P) + pl - d_l(t)\} + \\ + F_{\alpha_{\nu_i}}(t - \sum_{i-1}(P) - pl).$$

Пусть  $i^* := \arg \min_{i=1,2,\dots,g+1} \{F(\pi^i)\}$ , тогда  $\pi_l(t) := \pi^{i^*}$ ,

$$F_l(t) := F(\pi^{i^*}),$$

$$G_l(t) := \{ \langle m, \sum_{i^*-1}(f) + \max\{0, \sum_{i^*-1}(P) + pl - d_l(t)\}, \\ \sum_{i^*-1}(P) + pl, (\pi_1, \dots, \pi_{i^*-1}, l) \rangle,$$

$$G_{\alpha_{\nu_{i^*}}}(t - \sum_{i^*-1}(P) - pl) \}.$$

2. Алгоритм возвращает расписание  $\pi_1(d_n)$  и соответствующее значение суммарного запаздывания  $F_1(d_n)$ .

**Теорема 5.** Расписание  $\pi_1(d_n)$  является оптимальным расписанием для случая (6). **Алгоритм В-к** строит расписание  $\pi_1(d_n)$  за  $O(kn \sum p_j)$  операций.

**Доказательство.** Каждый шаг алгоритма **В-к** организован в соответствии с леммами 16 и 17. Пусть  $G(\pi_{l+1}(t)) = (\pi_1, \dots, \pi_g)$ . При работе алгоритма требование  $l$  ставится на обслуживание на следующие позиции: перед всеми требованиями множества  $N_{l+1}$ , после  $\pi_1$ , после  $\pi_2$  и т.д. Эти возможные позиции просматриваются в ходе работы алгоритма путем перебора элементов множеств  $G_{l+1}(t)$  и выбирается наилучшее из получившихся расписаний. Постановка требования  $l$  "внутри" любого  $\pi_i$  приведет к

тому, что  $G(\pi_l(t)) = \emptyset$ . Осуществляя перебор таким образом, построенные расписания  $\pi_l(t)$  для любых  $l$  и  $t$  будут оптимальными для набора требований  $N_l$  с директивными сроками  $d_j(t)$ . Следовательно расписание  $\pi_1(d_n)$  является оптимальным для случая (6).

При работе алгоритма достаточно исследовать только целочисленные точки в интервале  $[t_0 + p_n, t_0 + 2 \sum_{j=1}^n p_j]$ , так как  $p_j \in \mathbb{Z}^+$ ,  $j \in N$ , и вне этого интервала оптимальные расписания известны (SPT и EDD последовательности), которые могут быть построены до работы алгоритма. Шаг 0 алгоритма выполняется за  $O(\sum p_j)$  операций. Основной шаг 1 выполняется  $n$  раз. На каждой итерации шага 1 рассматривается не более чем  $k \leq n$  позиций для требования  $l$ , выбирается наилучшая позиция и вычисляются  $G_l(t)$  и  $F_l(t)$ . Для каждой точки  $t$  эти операции выполняются за  $O(k)$  операций. Таким образом, основной шаг алгоритма выполняется за  $O(k \sum p_j)$  операций. Следовательно, **алгоритм В-k** выполняется за  $O(kn \sum p_j)$  операций. Теорема доказана.  $\square$

Заметим, что если  $d_n \notin \mathbb{Z}$ , то построим и решим, по аналогии со случаем  $k = 1$ , пример  $\langle \{p_j, d'_j\}_{j \in N}, t'_0 \rangle$ , где  $d'_j = d_j - \Delta$ ,  $j \in N$ ,  $t'_0 = t_0 - \Delta$ ,  $\Delta = d_n - \lfloor d_n \rfloor$ .

Также необходимо заметить, что **алгоритмы В-1** и **В-k** находят оптимальные расписания  $\pi_1(t)$  для любого целочисленного  $t$ , т.е. "попутно" решается гораздо более общая задача, когда директивные сроки всех требований "двигаются" по временной оси.

### 3.7. Алгоритм решения в случае $k = n$

В случае  $k = n$  параметры требований удовлетворяют следующим условиям

$$d_j - d_{j-1} > p_j, \quad j = 2, 3, \dots, n, \quad (7)$$

без ограничений на продолжительность обслуживания требований и их целочисленность.

Для данного случая предлагается **Алгоритм В-н** трудоёмкости  $O(n^2)$  построения оптимального расписания.

#### Процедура **ProcB-n** ( $N, t$ )

0. Дано множество требований  $N = \{1, 2, \dots, n\}$ , момент начала обслуживания  $t$ ,  $d_1 < d_2 < \dots < d_n$ ;
1.  $S_\alpha := t + p_1 + p_2 + \dots + p_\alpha$ ,  $\alpha = 1, 2, \dots, n$ ;  
 $j^* = \arg \max_{j \in N} \{d_j : p_j = \max_{i \in N} p_i\}$ ;
2.  $\alpha^* := \arg \min_{\alpha=1, \dots, n} \{d_j + p_j \leq S_\alpha < d_{\alpha+1}, j \in \{j^* + 1, \dots, \alpha\}\}$ ;
3.  $N' := \{1, \dots, \alpha^*\} \setminus \{j^*\}$ ,  $t' := t$ ;  
 $N'' := \{\alpha^* + 1, \dots, n\}$ ,  $t'' := S_{\alpha^*}$ ;
4. Оптимальное расписание  $\pi^* := (\pi', j^*, \pi'')$ , где частичные расписания  $\pi', \pi''$  находятся рекурсивно  $\pi' := \mathbf{ProcB-n}(N', t')$ ,  $\pi'' := \mathbf{ProcB-n}(N'', t'')$ .

#### Алгоритм **В-н**

$$\pi^* := \mathbf{ProcB-n}(N, t_0)$$

Рассмотрим некоторый пример (подпример)  $\{N', t'\}$ . Пусть множество подходящих позиций для требования  $j^*$   $L(N', t') = \{\alpha_1, \dots, \alpha_m\}$ .

**Лемма 18.** Если параметры требований  $d_j, p_j, j \in N'$ , удовлетворяют условиям (7), то существует оптимальное расписание обслуживания требований  $\pi^*$ , при котором требование  $j^*$  обслуживается на позиции  $\alpha_1$ .

**Доказательство.** Рассмотрим две соседние позиции  $\alpha_i$  и  $\alpha_{i+1}$  из множества  $L(N', t')$ , где требование  $j^* = \arg \max_{j \in N'} \{d_j : p_j = \max_{i \in N'} p_i\}$ . Ниже будем обозначать  $\alpha = \alpha_i$  и  $\beta = \alpha_{i+1}$ .

Обозначим через  $\pi_\alpha$  и  $\pi_\beta$  расписания, при которых требование  $j^*$  обслуживается на позициях  $\alpha$  и  $\beta$ , соответственно. То есть,  $\pi_\alpha = (\pi_1, j^*, \pi_2)$ , где  $\pi_1$  есть оптимальное расписание обслуживания требований множества  $\{1, \dots, \alpha\} \setminus \{j^*\}$  с момента времени  $t'$ , и  $\pi_2$  есть оптимальное расписание обслуживания требований множества  $\{\alpha + 1, \dots, n\}$  с момента времени  $S_\alpha = t' + \sum_{i=1}^{\alpha} p_i$ . Аналогично  $\pi_\beta = (\pi'_1, j^*, \pi'_2)$ , где  $\pi'_1$  есть оптимальное расписание обслуживания требований множества  $\{1, \dots, \beta\} \setminus \{j^*\}$  с момента времени  $t'$  и  $\pi'_2$  есть оптимальное расписание обслуживания требований множества  $\{\beta + 1, \dots, n\}$  с момента времени  $S_\beta = t' + \sum_{i=1}^{\beta} p_i$ .

Рассмотрим случай  $d_{j^*} \geq S_\alpha$ . С одной стороны, выполняются условия  $S_\alpha \leq d_{j^*} < d_{\alpha+1}$  и  $d_\beta + p_\beta \leq S_\beta$ . Следовательно,  $d_\beta - d_{j^*} < S_\beta - S_\alpha$ . С другой стороны, вследствие условий (7), мы имеем следующее неравенство  $d_\beta - d_{j^*} \geq p_{\alpha+1} + p_{\alpha+2} + \dots + p_\beta = S_\beta - S_\alpha$ . Получаем противоречие и  $\beta \notin L(N', t')$ , множество  $L(N', t')$  содержит только одну позицию  $\alpha_1$  для требования  $j^*$ .

Рассмотрим случай, когда  $d_{j^*} < S_\alpha$ . Покажем, что выполняется  $\pi'_1 = (\pi_1, \bar{\pi})$ , где  $\bar{\pi} = (\alpha + 1, \alpha + 2, \dots, \beta)$ . Имеем  $S_\alpha - p_{j^*} + p_{\alpha+1} < S_\alpha < d_{\alpha+1}$ . Следовательно, будем иметь  $T_{\alpha+1}(\bar{\pi}, S_\alpha - p_{j^*}) = 0$ . Из условия (7) следует, что  $T_j(\bar{\pi}, S_\alpha - p_{j^*}) = 0$  для всех  $j \in \{\bar{\pi}\}$ . Получаем  $\pi'_1 = (\pi_1, \bar{\pi})$ .

Рассмотрим расписание  $\pi' = (\pi_1, j^*, \bar{\pi}', \pi'_2)$ , где

$\bar{\pi}' = (\alpha + 2, \alpha + 3, \dots, \beta, \alpha + 1)$ . Так как  $\pi_\alpha$  является расписанием с наименьшим значением суммарного запаздывания среди множества расписаний, при которых требование  $j^*$  обслуживается на позиции  $\alpha$ , то имеем  $F(\pi_\alpha, t') \leq F(\pi', t')$ .

Покажем, что выполняется  $F(\pi', t') < F(\pi_\beta, t')$ . Так как для всех  $j \in \{\bar{\pi}\}$  выполняется  $T_j(\bar{\pi}, S_\alpha - p_{j^*}) = 0$ , то

$$F(\pi', t') - F(\pi_\beta, t') = \Delta + F(\bar{\pi}', S_\alpha),$$

где  $\Delta = T_{j^*}(\pi', t') - T_{j^*}(\pi_\beta, t')$ . Так как  $d_{j^*} < S_\alpha$ , то требование  $j^*$  запаздывает при обоих расписаниях  $\pi'$  и  $\pi_\beta$ , и величина  $\Delta = -\sum_{j=\alpha+1}^{\beta} p_j = S_\alpha - S_\beta$ .

Вычислим величину  $F(\bar{\pi}', S_\alpha)$ . Из условий (7) и  $S_\alpha < d_{\alpha+1}$  следует  $T_j(\bar{\pi}', S_\alpha) = 0$  для всех  $j \in \{\alpha + 2, \dots, \beta\}$ , и из условия  $d_{\alpha+1} + p_{\alpha+1} \leq S_\beta$  следует  $T_{\alpha+1}(\bar{\pi}', S_\alpha) = S_\beta - d_{\alpha+1}$ . Следовательно,  $F(\pi', t') - F(\pi_\beta, t') = S_\alpha - S_\beta + S_\beta - d_{\alpha+1} < 0$ .

Мы показали, что  $F(\pi_\alpha, t') \leq F(\pi', t') < F(\pi_\beta, t')$ . Поскольку  $\alpha$  и  $\beta$  произвольные соседние позиции для требования  $j^*$ , то выполняется

$$F(\pi_{\alpha_1}, t') < F(\pi_{\alpha_2}, t') < \dots < F(\pi_{\alpha_m}, t').$$

Таким образом, при любом оптимальном расписании обслуживания требований множества  $N'$  с момента времени  $t'$  требование  $j^*$  обслуживается на первой позиции из множества  $L(N', t')$ . Лемма доказана.  $\square$

**Теорема 6. Алгоритм В-п** строит оптимальное расписание для случая (7) за  $O(n^2)$  операций.

**Доказательство.** Как было показано в лемме 18, в случае (7) существует оптимальное расписание обслуживания требований множества  $N$  с момента времени  $t_0$ , при котором требование  $j^*$  обслуживается на первой

позиции из множества  $L(N, t_0)$ . Любой подпример  $\{N', t'\}$ ,  $N' \subseteq N$ ,  $t' \geq t_0$ , исходного примера, очевидно, также удовлетворяет случаю (7). Поэтому, **алгоритм В-н**, являющийся модификацией **алгоритма А** на случай перебора только первой позиции из множества  $L(N', t')$ , строит оптимальное расписание в случае (7).

Каждый вызов процедуры **ProcВ-н**( $N, t$ ) фиксирует позицию одного требования в оптимальном расписании  $\pi^*$  и выполняется за не более, чем  $O(n)$  операций. Следовательно, **алгоритм В-н** выполняется за  $O(n^2)$  операций. Теорема доказана.  $\square$

## 4. Канонические примеры проблемы $1 \parallel \sum T_j$

В этом разделе будут описаны два NP-трудных случая проблемы  $1 \parallel \sum T_j$  – канонические примеры DL[5] и LG. NP-трудность канонических примеров доказана сведением NP-полной проблемы Четно-Нечетного Разбиения к проблеме  $1 \parallel \sum T_j$ .

### 4.1. Проблема Четно-Нечетного Разбиения (ЧНР).

Задано упорядоченное множество из  $2n$  положительных целых чисел  $B = \{b_1, b_2, \dots, b_{2n}\}$ ,  $b_i > b_{i+1}$ ,  $1 \leq i \leq 2n - 1$ . Требуется определить, существует ли разбиение множества  $B$  на два подмножества  $B_1$  и  $B_2$ , такое, что  $\sum_{b_i \in B_1} b_i = \sum_{b_i \in B_2} b_i$  и для каждого  $i = 1, 2, \dots, n$  подмножество  $B_1$  (следовательно, и  $B_2$ ) содержит в точности один элемент из пары  $\{b_{2i-1}, b_{2i}\}$ .

Обозначим  $\delta_i = b_{2i-1} - b_{2i}$ ,  $i = 1, \dots, n$ ,  $\delta = \sum_{i=1}^n \delta_i$ .

Построим модифицированный пример **Четно-Нечетного Разбиения**.

$$\begin{cases} a_{2n} = M + b, \\ a_{2i} = a_{2i+2} + b, \quad i = n - 1, \dots, 1, \\ a_{2i-1} = a_{2i} + \delta_i, \quad i = n, \dots, 1, \end{cases} \quad (8)$$

где  $b \gg n\delta$ ,  $M \geq n^3b$ .

Очевидно выполняется  $a_i > a_{i+1}$ ,  $\forall i = 1, 2, \dots, 2n - 1$ ,  $\delta_i = b_{2i-1} - b_{2i} = a_{2i-1} - a_{2i}$ ,  $i = 1, \dots, n$ .

**Лемма 19.** Исходный пример **ЧНР** имеет решение "ДА" тогда и только тогда, когда модифицированный пример **ЧНР** имеет решение "ДА".

**Доказательство.** Пусть для исходного примера существует два подмножества  $B_1$  и  $B_2$ , таких что  $\sum_{b_i \in B_1} b_i =$

$\sum_{b_i \in B_2} b_i$ . Возьмем  $A_1 = \{a_i | b_i \in B_1\}$ ,  $A_2 = \{a_i | b_i \in B_2\}$ . Тогда выполняется  $\sum_{a_i \in A_1} a_i = \sum_{a_i \in A_2} a_i$ .

Пусть для модифицированного примера существует два подмножества  $A_1$  и  $A_2$ , таких что  $\sum_{a_i \in A_1} a_i = \sum_{a_i \in A_2} a_i$ . Возьмем  $B_1 = \{b_i | a_i \in A_1\}$ ,  $B_2 = \{b_i | a_i \in A_2\}$ . Очевидно,  $\sum_{b_i \in B_1} b_i = \sum_{b_i \in B_2} b_i$ .  $\square$

## 4.2. Канонические DL-примеры [5] проблемы $1 || \sum T_j$

Приведем полиномиальную схему сведения **ЧНР** к проблеме  $1 || \sum T_j$  [5]. Определим  $\delta = \sum_{i=1}^n (b_{2i-1} - b_{2i})$ .

Пусть  $a_{2i-1} = b_{2i-1} + (9n^2 + 3n - i + 1)\frac{1}{2}\delta + 5n(b_1 - b_{2n})$  и  $a_{2i} = b_{2i} + (9n^2 + 3n - i + 1)\frac{1}{2}\delta + 5n(b_1 - b_{2n})$ ,  $i = 1, \dots, n$ .

Построим *канонический пример* [5] проблемы

$1 || \sum T_j$  для множества из  $3n + 1$  требований  $N = \{V_1, V_2, \dots, V_{2n}, W_1, W_2, \dots, W_{n+1}\}$ . Пусть  $b = (4n + 1)\frac{1}{2}\delta$ . Зададим параметры требований следующим образом:

$$\begin{aligned} p_{V_i} &= a_i, & i &= 1, 2, \dots, 2n, \\ p_{W_i} &= b, & i &= 1, 2, \dots, n + 1, \\ d_{V_i} &= \begin{cases} (j-1)b + \frac{1}{2}\delta + (a_2 + a_4 + \dots + a_{2i}), & \text{если } i = 2j - 1, \\ d_{V_{2j-1}} + 2(n-j+1)(a_{2j-1} - a_{2j}), & \text{если } i = 2j, \end{cases} \\ & & & j = 1, 2, \dots, n, \\ d_{W_i} &= \begin{cases} ib + (a_2 + a_4 + \dots + a_{2i}), & \text{если } i = 1, 2, \dots, n, \\ d_{W_n} + \frac{1}{2}\delta + b, & \text{если } i = n + 1. \end{cases} \end{aligned}$$

Пусть  $\{V_{i,1}, V_{i,2}\} = \{V_{2i-1}, V_{2i}\}$ ,  $i = 1, \dots, n$ . Определим *каноническое расписание* как расписание вида

$$\pi = (V_{1,1}, W_1, V_{2,1}, W_2, \dots, W_{n-1}, V_{n,1}, W_n, W_{n+1}, V_{n,2}, V_{n-1,2}, \dots, V_{1,2}).$$

**Теорема 7.** [5] Для примеров канонической задачи проблемы  $1 || \sum T_j$ , к которой сводится **ЧНР**, существует оптимальное расписание, которое является каноническим.  $\square$

Если  $\frac{1}{2}\delta \notin Z$ , рассмотрим исходный пример **ЧНР**, где все  $b_i$  умножены на 2. Несложно показать эквивалентность исходного и нового примера.

Введем следующие обозначения:  $d_j(t) = d_j - d_{W_{n+1}} + t$ ,  $j \in N$ ,  $t \in R$ . Пусть  $\pi_l(t)$  и  $F_l(t)$  оптимальное расписание и соответствующее ему значение суммарного запаздывания для параметрического примера с множеством из  $3(n-l) + 1$  требований  $N_l = \{V_{2l-1}, V_{2l}, W_l, \dots, V_{2n-1}, V_{2n}, W_n, W_{n+1}\}$  и

директивными сроками окончания обслуживания  $d_j(t)$ ,  $j = V_{2l-1}, V_{2l}, W_l, \dots, V_{2n-1}, V_{2n}, W_n, W_{n+1}$ ,  $l = n, \dots, 1$ .

### Алгоритм В-1 канонический

0.  $\pi_{n+1}(t) := (W_{n+1})$ ,  $F_{n+1}(t) := \max\{0, b - t\}$  для  
 $t \in T_{n+1} := [d_{W_{n+1}} - \sum_{i=1}^n a_{2i} - nb - \delta, d_{W_{n+1}} - \sum_{i=1}^n a_{2i} - nb]$

1. **FOR**  $l = n, n - 1, \dots, 1$ , для  
 $t \in T_l := [d_{W_{n+1}} - \sum_{i=1}^{l-1} a_{2i} - (l-1)b -$   
 $(\delta - \sum_{i=l-1}^n \delta_i), d_{W_{n+1}} - \sum_{i=1}^{l-1} a_{2i} - (l-1)b]:$

$$\pi^1 := (V_{2l-1}, W_l, \pi_{l+1}(t - a_{2l-1} - b), V_{2l}),$$

$$\pi^2 := (V_{2l}, W_l, \pi_{l+1}(t - a_{2l} - b), V_{2l-1});$$

$$F(\pi^1) := \max\{0, a_{2l-1} - d_{V_{2l-1}}(t)\} +$$

$$+ \max\{0, a_{2l-1} + b - d_{W_l}(t)\} + F_{l+1}(t - a_{2l-1} - b) +$$

$$+ \max\{0, \sum_{j=l}^n (a_{2j-1} + a_{2j} + b) + b - d_{V_{2l}}(t)\};$$

$$F(\pi^2) := \max\{0, a_{2l} - d_{V_{2l}}(t)\} +$$

$$+ \max\{0, a_{2l} + b - d_{W_l}(t)\} + F_{l+1}(t - a_{2l} - b) +$$

$$+ \max\{0, \sum_{j=l}^n (a_{2j-1} + a_{2j} + b) + b - d_{V_{2l-1}}(t)\};$$

$$F_l(t) := \min\{F(\pi^1), F(\pi^2)\};$$

$$\pi_l(t) := \arg \min\{F(\pi^1), F(\pi^2)\}.$$

2. Алгоритм возвращает расписание  $\pi_1(d_{W_{n+1}})$  и соответствующее значение суммарного запаздывания  $F_1(d_{W_{n+1}})$ .

Оптимальное расписание для исходного примера будет равно  $\pi_1(d_{W_{n+1}})$ . Заметим, что шаги основного цикла алгоритма выполняются только для каждого целого  $t$  из интервала, длина которого не превышает  $\delta$ .

**Теорема 8.** Алгоритм **В-1 канонический** строит оптимальное каноническое расписание за время  $O(n\delta)$  для канонических примеров проблемы  $\mathbf{1} \parallel \sum T_j$ , к которым сводятся примеры **ЧНР**.  $\square$

Необходимо отметить, что все семейство алгоритмов В-1 находит множество оптимальных расписаний для задач  $\langle \{p_j, d_j(t)\}, 0 \rangle$ , когда  $t$  "пробегают" по временной оси, находится множество оптимальных расписаний для разных значений директивных сроков.

### 4.3. Канонические LG-примеры проблемы $1 || \sum T_j$

Показано, что частный случай **В-1** (2) [11] задачи минимизация суммарного запаздывания для одного прибора  $1 || \sum T_j$  является NP-трудным в обычном смысле.

Приведем полиномиальную схему сведения модифицированного примера **ЧНР** к частному случаю (2) задачи  $1 || \sum T_j$ .

Количество требований  $2n + 1$ . Требования обозначим следующим образом:

$$V_1, V_2, V_3, V_4, \dots, V_{2i-1}, V_{2i}, \dots, V_{2n-1}, V_{2n}, V_{2n+1},$$

$N = \{1, 2, \dots, 2n, 2n + 1\}$ . Для упрощения записи будем использовать следующие обозначения  $p_{V_i} = p_i$ ,  $d_{V_i} = d_i$ ,  $T_{V_i} = T_i$ ,  $c_{V_i} = c_i$ ,  $i = 1, \dots, 2n + 1$ . Пример, удовлетворяющий следующим ограничениям, будем называть *каноническим LG-примером*.

$$\left\{ \begin{array}{ll} p_1 > p_2 > \dots > p_{2n+1}, & (9.1) \\ d_1 < d_2 < \dots < d_{2n+1}, & (9.2) \\ d_{2n+1} - d_1 < p_{2n+1}, & (9.3) \\ p_{2n+1} = M = n^3 b, & (9.4) \\ p_{2n} = p_{2n+1} + b = a_{2n}, & (9.5) \\ p_{2i} = p_{2i+2} + b = a_{2i}, \quad i = n - 1, \dots, 1, & (9.6) \\ p_{2i-1} = p_{2i} + \delta_i = a_{2i-1}, \quad i = n, \dots, 1, & (9.7) \\ d_{2n+1} = \sum_{i=1}^n p_{2i} + p_{2n+1} + \frac{1}{2}\delta, & (9.8) \\ d_{2n} = d_{2n+1} - \delta, & (9.9) \\ d_{2i} = d_{2i+2} - (n - i)b + \delta, \quad i = n - 1, \dots, 1, & (9.10) \\ d_{2i-1} = d_{2i} - (n - i)\delta_i - \varepsilon\delta_i, \quad i = n, \dots, 1, & (9.11) \end{array} \right. \quad (9)$$

где  $b = n^2\delta$ ,  $0 < \varepsilon < \frac{\min_i \delta_i}{\max_i \delta_i}$ .

Директивные сроки примера (9) представлены на рис. 2.

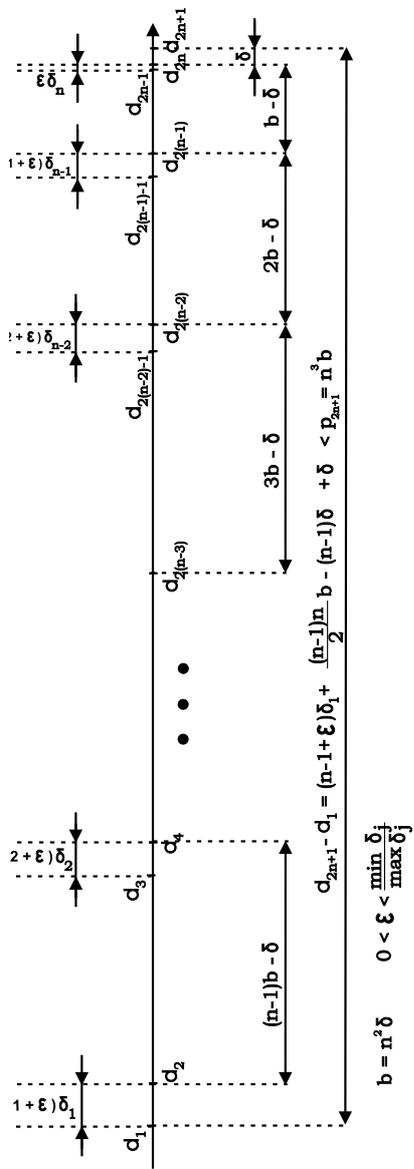


Рис. 2. Директивные сроки канонического LG-примера.

Обозначим через  $L = \frac{1}{2} \sum_{i=1}^{2n} p_i$ , тогда  $d_{2n+1} = L + p_{2n+1}$ , так как  $\frac{1}{2} \sum_{i=1}^{2n} p_i = \sum_{i=1}^n p_{2i} + \frac{1}{2}\delta$ .

Необходимо отметить, что канонические примеры DL из работы [5] не удовлетворяют случаю (9). Первые три неравенства (9.1) - (9.3) показывают, что (9) является подслучаем (2).

#### 4.4. Свойства примеров случая (9) проблемы $1 \parallel \sum T_j$

**Лемма 20.** Для случая (9) при любом расписании количество запаздывающих требований равно или  $n$ , или  $(n + 1)$ .

**Доказательство.**

1. Рассмотрим подмножество требований  $N'$  состоящее из  $(n + 2)$  самых коротких по продолжительности обслуживания требований и упорядочим их вначале расписания. Очевидно, что  $\sum_{i \in N'} p_i > (n + 2)p_{min} = (n + 2)n^3b$ , где  $p_{min} = \min_{j \in N} \{p_j\} = p_{2n+1}$ .

Из (9.4)-(9.8) найдем

$$d_{max} = \max_{j \in N} \{d_j\} = d_{2n+1} = (n+1)n^3b + (b+2b+\dots+nb) + \frac{1}{2}\delta,$$

для которого будет выполняться

$$d_{max} = d_{2n+1} = (n + 1)n^3b + \frac{n(n + 1)}{2}b + \frac{1}{2}\delta < (n + 2)n^3b < \sum_{i \in N'} p_i,$$

т.е. требование обслуживаемое  $(n+2)$  по порядку будет запаздывать при любом расписании. Все последующие требования будут также запаздывать, так как согласно (9.3) разность между директивными сроками любых двух требований меньше продолжительности обслуживания каждого требования. Таким образом, при любом расписании  $\pi$  количество незапаздывающих требований не превышает  $(n+1)$ .

2. Рассмотрим подмножество  $N''$  состоящее из  $n$  самых длинных по продолжительности обслуживания требований и упорядочим их вначале расписания. Возможны два случая:

- а) пусть  $n = 2k$ , тогда  $N'' = \{V_1, V_2, \dots, V_{2k-1}, V_{2k}\}$ , будем иметь

$$P(N'') = nn^3b + 2(nb + (n-1)b + \dots + (n-k+1)b) + \sum_{i=1}^k \delta_i,$$

$$P(N'') = nn^3b + 2\left(\frac{n(n+1)}{2} - \frac{(n-k)(n-k+1)}{2}\right)b + \sum_{i=1}^k \delta_i.$$

Из (9.8) - (9.11) будем иметь

$$d_{min} = \min_{j \in N} \{d_j\} = d_1 = d_{2n+1} - \left(\sum_{i=1}^{n-1} ((n-i)b - \delta) + \delta + (n-1)\delta_1 - \varepsilon\delta_1\right) = (n+1)n^3b + (b+2b+\dots+nb) + \frac{1}{2}\delta - \left(\sum_{i=1}^{n-1} ((n-i)b - \delta) + \delta + (n-1)\delta_1 + \varepsilon\delta_1\right) > P(N'');$$

б) пусть  $n = 2k + 1$ , тогда

$$N'' = \{V_1, V_2, \dots, V_{2k-1}, V_{2k}, V_{2(k+1)-1}\}.$$

$$P(N'') = nn^3b + 2(nb + (n-1)b + \dots + (n-k+1)b) + \\ + (n-k)b + \sum_{i=1}^{k+1} \delta_i,$$

$$P(N'') = nn^3b + 2\left(\frac{n(n+1)}{2} - \frac{(n-k)(n-k+1)}{2}\right)b + \\ + (n-k)b + \sum_{i=1}^{k+1} \delta_i,$$

$$d_{min} = d_1 = d_{2n+1} - \left(\sum_{i=1}^{n-1} ((n-i)b - \delta) + \delta + (n-1)\delta_1 - \varepsilon\delta_1\right) = (n+1)n^3b + (b + 2b + \dots + nb) + \frac{1}{2}\delta - \\ - \left(\sum_{i=1}^{n-1} ((n-i)b - \delta) + \delta + (n-1)\delta_1 + \varepsilon\delta_1\right) > P(N''),$$

т.е. при любом расписании первые  $n$  требований будут не запаздывать. Таким образом, при любом расписании  $\pi$  количество запаздывающих требований больше или равно  $n$ .

Следовательно, для случая (9) при любом расписании количество запаздывающих требований равно или  $n$ , или  $(n+1)$ .  $\square$

**Лемма 21.** Для случая (9), для каждого расписания  $\pi = (\pi_1, \pi_2)$  существует расписание вида  $\pi' = (\pi_{EDD}, \pi_{SPT})$ , где  $\{\pi_1\} = \{\pi_{EDD}\}$ ,  $\{\pi_2\} = \{\pi_{SPT}\}$ ,  $|\{\pi_1\}| = n + 1$ ,  $|\{\pi_2\}| = n$ . Причем,  $F(\pi) \geq F(\pi')$ .

**Доказательство.**

Рассмотрим частичное расписание  $\pi_1$ . Так как  $n$  первых требований при  $\pi_1$  не запаздывают, а может запаздывать лишь последнее требование, то EDD порядок обслуживания требований множества  $\{\pi_1\}$  будет оптимальным для этого подмножества. В этом случае на  $(n + 1)$  месте будет обслуживаться требование  $j = \operatorname{argmax}\{d_i : i \in \{\pi_1\}\}$ .

Рассмотрим частичное расписание  $\pi_2$ . Так как все  $n$  требований при  $\pi_2$  запаздывают, то порядок SPT обслуживания требований множества  $\{\pi_2\}$  будет оптимальным.  $\square$

Расписание вида

$$(V_{1,1}, V_{2,1}, \dots, V_{i,1}, \dots, V_{n,1}, V_{2n+1}, V_{n,2}, \dots, V_{i,2}, \dots, V_{2,2}, V_{1,2})$$

будем называть *каноническим LG-расписанием*, где  $\{V_{i,1}, V_{i,2}\} = \{V_{2i-1}, V_{2i}\}$ ,  $i = 1, 2, \dots, n$ .

**Лемма 22.** Если расписание  $\pi = (\pi_1, \pi_2)$ , для которого  $|\{\pi_1\}| = (n + 1)$ ,  $|\{\pi_2\}| = n$ , неканоническое LG или не может быть преобразовано к каноническому LG расписанию применением правил EDD и SPT к частичным расписаниям  $\pi_1$  и  $\pi_2$ , соответственно, то при расписании  $\pi$  некоторая пара требований  $\{V_{2i-1}, V_{2i}\}$ ,  $i < n$ , не запаздывает или расписание  $\pi$  имеет вид

$$(V_{1,1}, \dots, V_{i,1}, \dots, V_{n-1,1}, V_{2n-1}, V_{2n}, V_{2n+1}, V_{n-1,2}, \dots, V_{i,2}, \dots, V_{1,2}), \quad (10)$$

т.е. пара требований  $\{V_{2n-1}, V_{2n}\}$  обслуживается до требования  $V_{2n+1}$ , одно требование из каждой пары  $\{V_{2i-1}, V_{2i}\}$ ,  $i = 1, \dots, n-1$ , обслуживается до требования  $V_{2n+1}$ , другое требование из пары после требования  $V_{2n+1}$ , а требования  $V_{2n+1}$  обслуживается  $(n+2)$  по порядку.

**Доказательство.** Рассмотрим некоторое расписание  $\pi = (\pi_1, \pi_2)$ , где  $|\{\pi_1\}| = (n+1)$ ,  $|\{\pi_2\}| = n$ . Рассмотрим возможные случаи.

1. Если  $\{\pi_2\} = \{V_{1,2}, \dots, V_{n,2}\}$ , т.е. при  $\pi_2$  упорядочены  $n$  требований по одному требованию из всех  $n$  пар  $\{V_{2i-1}, V_{2i}\}$ ,  $i = 1, \dots, n$ . Упорядочим требования из  $\pi_2$  по правилу SPT, а требования из  $\pi_1$  по правилу EDD. Тогда получим каноническое расписание  $\pi'$ , причем  $F(\pi') \leq F(\pi)$ , согласно лемме 21.
2. Если  $\{\pi_2\} \neq \{V_{1,2}, \dots, V_{n,2}\}$ , то возможны ситуации.
  - а)  $V_{2n+1} \in \{\pi_2\}$ ,
  - б) существует пара требований  $\{V_{2j-1}, V_{2j}\} \subset \{\pi_2\}$ .

Тогда, учитывая что  $|\{\pi_2\}| = n$ , для некоторого  $i$  будем иметь  $\{V_{2i-1}, V_{2i}\} \subset \{\pi_1\}$ .  $\square$

Далее в теореме 9 будет показано, что для случая (9) все оптимальные расписания являются каноническими LG. Доказано, что произвольное неканоническое LG-расписание  $\pi$  может быть преобразовано к каноническому LG-расписанию  $\pi'$ , причем  $F(\pi) > F(\pi')$ . При доказательстве теоремы используются выводы лемм 23 - 26.

**Лемма 23.** Пусть расписание  $\pi$  имеет вид (10), при котором требование  $V_{2n+1}$  обслуживается на  $(n + 2)$  по порядку месте. Тогда для канонического LG-расписания  $\pi' = (V_{1,1}, \dots, V_{i,1}, \dots, V_{n-1,1}, V_{2n-1}, V_{2n+1}, V_{2n}, V_{n-1,2}, \dots, V_{i,2}, \dots, V_{1,2})$  выполняется  $F(\pi) > F(\pi')$ .

**Доказательство.** При расписании  $\pi$  требование  $V_{2n-1}$  обслуживается на позиции  $n$  "слева". Согласно лемме 4 требование  $V_{2n-1}$  не запаздывает, так как оно обслуживается  $n$ -ым по порядку. Требование  $V_{2n+1}$  обслуживается на позиции  $n + 2$  "слева" поэтому запаздывает.

Для требований множества  $\{V_2, V_4, \dots, V_{2i}, \dots, V_{2n-2}, V_{2n-1}\}$  имеем

$$\begin{aligned} P(\{V_2, V_4, \dots, V_{2i}, \dots, V_{2n-2}, V_{2n-1}\}) &= nn^3b + \sum_{k=1}^n kb + \delta_n = \\ &= d_{2n+1} - n^3b - \frac{1}{2}\delta + \delta_n, \end{aligned}$$

согласно (9.8). Далее, очевидно,

$$\begin{aligned} P(\{V_{1,1}, V_{2,1}, \dots, V_{i,1}, \dots, V_{n-1,1}, V_{2n-1}\}) + p_{2n} &\geq \\ &\geq P(\{V_2, V_4, \dots, V_{2i}, \dots, V_{2n-2}, V_{2n-1}\}) + p_{2n}, \end{aligned}$$

поэтому

$$c_{2n}(\pi) \geq d_{2n+1} + b - \frac{1}{2}\delta + \delta_n > d_{2n}.$$

Следовательно, требование  $V_{2n}$ , которое обслуживается  $(n + 1)$  по порядку, при расписании  $\pi$  запаздывает.

Расписание  $\pi$  можно записать как  $\pi = (\pi_{11}, V_{2n}, V_{2n+1}, \pi_{21})$ . Рассмотрим следующее каноническое LG-расписание  $\pi' = (\pi_{11}, V_{2n+1}, V_{2n}, \pi_{21})$ . Покажем, что  $F(\pi) > F(\pi')$ .

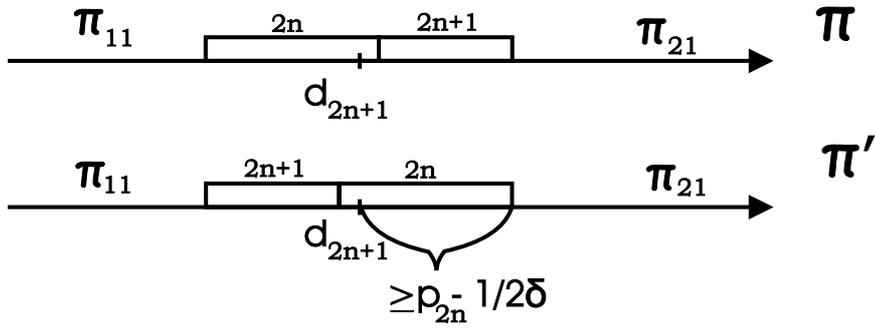


Рис. 3. Перестановка требований  $V_{2n}$  и  $V_{2n+1}$

- а) Пусть при расписании  $\pi'$  требование  $V_{2n+1}$  не запаздывает. Тогда из (9.8) следует, что  $d_{2n+1} - c_{2n+1}(\pi') \leq \frac{1}{2}\delta$ , так как расписание  $\pi'$  каноническое.

Из рис. 3 видно, что

$$\begin{aligned}
 F(\pi) - F(\pi') &= T_{2n}(\pi) + T_{2n+1}(\pi) - (T_{2n}(\pi') + T_{2n+1}(\pi')) = \\
 &= (T_{2n+1}(\pi) - T_{2n+1}(\pi')) - (T_{2n}(\pi') - T_{2n}(\pi)) \geq (p_{2n} - \frac{1}{2}\delta) - \\
 &= p_{2n+1} = p_{2n+1} + b - \frac{1}{2}\delta - p_{2n+1} > 0.
 \end{aligned}$$

- б) Пусть при расписании  $\pi'$  требование  $V_{2n+1}$  запаздывает, тогда

$$\begin{aligned}
 F(\pi) - F(\pi') &= T_{2n}(\pi) + T_{2n+1}(\pi) - (T_{2n}(\pi') + T_{2n+1}(\pi')) = \\
 &= p_{2n} - p_{2n+1} = b > 0. \square
 \end{aligned}$$

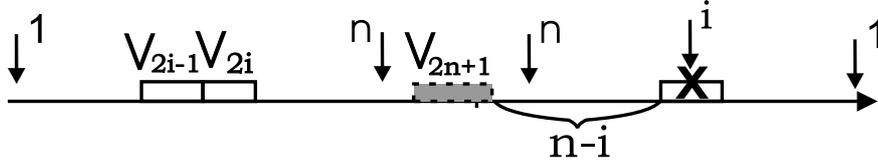


Рис. 4. Перестановка в неканоническом расписании

**Лемма 24.** Пусть при некотором расписании  $\pi = (\pi_{11}, V_{2i-1}, V_{2i}, \pi_{12}, \pi_{21}, X, \pi_{22})$  пара требований  $\{V_{2i-1}, V_{2i}\}$ ,  $i < n$ , не запаздывает, а на позиции  $i$  "справа" обслуживается требование  $X \in \{V_{2j-1}, V_{2j}\}$ ,  $j \geq i + 1$ . Тогда для расписания  $\pi' = (\pi_{11}, V_{2i-1}, X, \pi_{12}, \pi_{21}, V_{2i}, \pi_{22})$  выполняется  $F(\pi) > F(\pi')$ .

**Доказательство.**

Пусть при расписании  $\pi$  запаздывают требования только из множества  $\{\pi_{21}, X, \pi_{22}\}$ , где  $|\{\pi_{22}\}| = (i - 1)$ . Требование  $X$  занимает позицию  $i$  "справа" (см. рис. 4), в которой при каноническом расписании будет обслуживаться требование  $V_{i,2} \in \{V_{2i-1}, V_{2i}\}$ .

Построим расписание  $\pi' = (\pi_{11}, V_{2i-1}, X, \pi_{12}, \pi_{21}, V_{2i}, \pi_{22})$ . При обоих расписаниях запаздывает не меньше  $n$  требований. Поэтому при расписании  $\pi'$  перед требованием  $V_{2i}$  будет запаздывать не меньше, чем  $n - i$  требований (и не больше, чем  $n - i + 1$  требований), согласно лемме 20. Поэтому

$$F(\pi) - F(\pi') \geq (p_{2i} - p_X)(n - i) - (d_X - d_{2i}).$$

а) Если  $X = V_{2j}$ , то  $p_{2i} - p_X = (j - i)b$ ,

$$d_X - d_{2i} = \sum_{k=i}^{j-1} (n - k)b - (j - i)\delta = n(j - i)b - \sum_{k=i}^{j-1} kb - (j - i)\delta = n(j - i)b - i(j - i)b - \sum_{k=0}^{j-1-i} kb - (j - i)\delta.$$

Следовательно,

$$F(\pi) - F(\pi') \geq (j-i)b(n-i) - (n(j-i)b - i(j-i)b - \sum_{k=0}^{j-1-i} kb - (j-i)\delta) = \sum_{k=0}^{j-1-i} kb + (j-i)\delta > 0.$$

б) Если  $X = V_{2j-1}$ , то  $p_{2i} - p_X = ((j-i)b - \delta_j)$ ,

$$\begin{aligned} d_X - d_{2i} &= \sum_{k=i}^{j-1} (n-k)b - (j-i)\delta - (n-j)\delta_j - \varepsilon\delta_j = \\ &= n(j-i)b - \sum_{k=i}^{j-1} kb - (j-i)\delta - (n-j)\delta_j - \varepsilon\delta_j = \\ &= n(j-i)b - i(j-i)b - \sum_{k=0}^{j-1-i} kb - (j-i)\delta - (n-j)\delta_j - \varepsilon\delta_j. \end{aligned}$$

Следовательно,

$$\begin{aligned} F(\pi) - F(\pi') &\geq ((j-i)b - \delta_j)(n-i) - (n(j-i)b - \\ &- i(j-i)b - \sum_{k=0}^{j-1-i} kb - (j-i)\delta - (n-j)\delta_j - \varepsilon\delta_j) = \\ &= \sum_{k=0}^{j-1-i} kb + (j-i)\delta - (j-i)\delta_j + \varepsilon\delta_j > 0. \square \end{aligned}$$

**Лемма 25.** Пусть при некотором расписании  $\pi = (\pi_{11}, V_{2i-1}, V_{2i}, \pi_{12}, \pi_{21}, X, \pi_{22})$  пара требований  $\{V_{2i-1}, V_{2i}\}$ ,  $i < n$ , не запаздывает, а на позиции  $i$  "справа" обслуживается требование  $X \in \{V_{2j-1}, V_{2j}\}$ ,  $j < i - 1$ . Тогда для расписания  $\pi' = (\pi_{11}, V_{2i-1}, X, \pi_{12}, \pi_{21}, V_{2i}, \pi_{22})$  выполняется  $F(\pi) > F(\pi')$ .

**Доказательство.**

Пусть при расписании  $\pi$  запаздывают требования только из множества  $\{\pi_{21}, X, \pi_{22}\}$ , где  $|\{\pi_{22}\}| = (i-1)$ . Требование  $X$  занимает позицию  $i$  "справа" (см. рис. 4), в которой при каноническом расписании будет обслуживаться требование  $V_{i,2} \in \{V_{2i-1}, V_{2i}\}$ .

Построим расписание  $\pi' = (\pi_{11}, V_{2i-1}, X, \pi_{12}, \pi_{21}, V_{2i}, \pi_{22})$ . При обоих расписаниях запаздывает не меньше  $n$  требований. Поэтому при расписании  $\pi'$  перед требованием  $V_{2i}$  будет запаздывать не меньше, чем  $(n-i)$  требований (и не больше, чем  $(n-i+1)$  требований), согласно лемме 20. Поэтому

$$F(\pi) - F(\pi') \geq (d_{2i} - d_X) - (p_X - p_{2i})(n - i + 1).$$

а) Если  $X = V_{2j}$ , то  $p_X - p_{2i} = (i - j)b$ ,

$$\begin{aligned} d_{2i} - d_{2j} &= \sum_{k=j}^{i-1} (n - k)b - (i - j)\delta = n(i - j)b - \sum_{k=j}^{i-1} kb - \\ &(i - j)\delta = n(i - j)b - (i - 1)(i - j)b + \sum_{k=0}^{i-1-j} kb - (i - j)\delta. \end{aligned}$$

Следовательно,

$$\begin{aligned} F(\pi) - F(\pi') &\geq n(i - j)b - (i - 1)(i - j)b + \sum_{k=0}^{i-1-j} kb - \\ &(i - j)\delta - (i - j)b(n - i + 1) = \sum_{k=0}^{i-1-j} kb - (i - j)\delta > 0. \end{aligned}$$

б) Если  $X = V_{2j-1}$ , то  $p_X - p_{2i} = (i - j)b + \delta_j$ ,

$$\begin{aligned} d_{2i} - d_{2j-1} &= \sum_{k=j}^{i-1} (n - k)b - (i - j)\delta + (n - j)\delta_j + \varepsilon\delta_j = \\ &n(i - j)b - \sum_{k=j}^{i-1} kb - (i - j)\delta + (n - j)\delta_j + \varepsilon\delta_j = \\ &n(i - j)b - (i - 1)(i - j)b + \sum_{k=0}^{i-1-j} kb - (i - j)\delta + (n - j)\delta_j + \varepsilon\delta_j. \end{aligned}$$

Следовательно,

$$\begin{aligned} F(\pi) - F(\pi') &\geq n(i - j)b - (i - 1)(i - j)b + \sum_{k=0}^{i-1-j} kb - \\ &- (i - j)\delta + (n - j)\delta_j + \varepsilon\delta_j - ((i - j)b + \delta_j)(n - i + 1) = \\ &\sum_{k=0}^{i-1-j} kb - (i - j)\delta - \delta_j + \varepsilon\delta_j > 0. \square \end{aligned}$$

**Лемма 26.** Пусть при некотором расписании  $\pi = (\pi_{11}, V_{2i-1}, V_{2i}, \pi_{12}, \pi_{21}, X, \pi_{22})$  пара требований  $\{V_{2i-1}, V_{2i}\}$ ,  $i < n$ , не запаздывает, а на позиции  $i$  "справа" обслуживается требование  $X \in \{V_{2(i-1)-1}, V_{2(i-1)}\}$ . Пусть при расписании  $\pi' = (\pi_{11}, V_{2i-1}, X, \pi_{12}, \pi_{21}, V_{2i}, \pi_{22})$  требование  $Y$  занимает позицию  $(n + 1)$  и  $T_Y(\pi') < 2\delta$ . Тогда выполняется  $F(\pi) > F(\pi')$ .

### Доказательство.

Пусть при расписании  $\pi$  запаздывают требования только из множества  $\{\pi_{21}, X, \pi_{22}\}$ , где  $|\{\pi_{22}\}| = (i - 1)$ . Требование  $X$  занимает позицию  $i$  "справа" (см. рис. 4), в которой при каноническом расписании будет обслуживаться требование  $V_{i,2} \in \{V_{2i-1}, V_{2i}\}$ .

Построим расписание  $\pi' = (\pi_{11}, V_{2i-1}, X, \pi_{12}, \pi_{21}, V_{2i}, \pi_{22})$ .

При расписании  $\pi$  перед требованием  $V_{2i}$  будет запаздывать не меньше  $(n - i)$  требований. Поэтому

$$\begin{aligned} F(\pi) - F(\pi') &> (d_{2i} - d_X) - (p_X - p_{2i})(n - i) - (T_Y(\pi') - T_Y(\pi)) > \\ &> (d_{2i} - d_X) - (p_X - p_{2i})(n - i) - 2\delta. \end{aligned}$$

Возможны ситуации:

- a) Если  $X = V_{2(i-1)}$ , то  $p_X - p_{2i} = b$ , то

$$d_{2i} - d_{2i-2} = (n - i + 1)b - \delta.$$

Поэтому

$$F(\pi) - F(\pi') > (n - i + 1)b - \delta - (n - i)b - 2\delta = b - 3\delta > 0.$$

- b) Если  $X = V_{2(i-1)-1}$ , то  $p_X - p_{2i} = b + \delta_{i-1}$ ,

$$d_{2i} - d_{2i-2} = (n - i + 1)b - \delta + (n - i + 1)\delta_{i-1} + \varepsilon\delta_{i-1}.$$

Следовательно,

$$\begin{aligned} F(\pi) - F(\pi') &> (n - i + 1)b - \delta + (n - i + 1)\delta_{i-1} + \varepsilon\delta_{i-1} - \\ &- (n - i)(b + \delta_{i-1}) - 2\delta = b - 3\delta + \delta_{i-1} + \varepsilon\delta_{i-1} > 0, \text{ так} \\ &\text{как } b = n^2\delta. \square \end{aligned}$$

Выводы леммы 26 используются при доказательстве теоремы 9. При этом имеет место  $T_Y(\pi') < 2\delta$ . Ситуация  $T_Y(\pi') \geq 2\delta$  нами не рассматривается, так как она не встречается.

На основе полученных лемм докажем следующую теорему.

**Теорема 9.** Для случая (9) все оптимальные расписания являются каноническими (или могут быть преобразованы к каноническим расписаниям применением правила EDD к  $(n+1)$  требованиям, обслуживаемым вначале расписания).

**Доказательство.**

Пусть  $\pi$  - произвольное расписание. Согласно лемме 21 можно рассматривать только расписание вида  $\pi = (\pi_{EDD}, \pi_{SPT})$ , где  $|\{\pi_{EDD}\}| = (n+1)$ . При этом расписании требование  $V_{2n+1}$  занимает или позицию  $(n+1)$ , или позицию  $(n+2)$ . Пусть расписание  $\pi$  не каноническое.

Так как  $\pi$  не является каноническим расписанием, то в силу леммы 22 имеем при расписании  $\pi$  пару требований  $\{V_{2i-1}, V_{2i}\}$ , которые не запаздывают,  $i < n$ , или расписание  $\pi$  имеет вид (10), при котором требование  $V_{2n+1}$  обслуживается на  $(n+2)$  месте.

Если расписание имеет вид (10), то по лемме 23 для канонического расписания  $\pi' = (V_{1,1}, \dots, V_{i,1}, \dots, V_{n-1,1}, V_{2n-1}, V_{2n+1}, V_{2n}, V_{n-1,2}, \dots, V_{i,2}, \dots, V_{1,2})$ , выполняется  $F(\pi) > F(\pi')$ . Переобозначим  $\pi := \pi'$ .

Далее опишем алгоритм, состоящий из двух циклов, преобразования исходного расписания  $\pi$  к каноническому виду.

**Цикл 1.** Пока среди не запаздывающих требований при очередном расписании  $\pi$  присутствует пара требований  $V_{2i-1}, V_{2i}$ , причем на позиции  $i$  "справа" обслуживается требование  $X \notin \{V_{2(i-1)-1}, V_{2(i-1)}\}$ ,  $X \neq V_{2n+1}$ . Применяем

для требований  $V_{2i}$  и  $X$  перестановку описанную в леммах 24 и 25. В результате значение целевой функции уменьшится.

**Конец цикла 1.** Переобозначим  $\pi := \pi'$ .

Количество шагов в цикле 1, очевидно, не превышает  $n$ .

Первые  $(n + 1)$  требований при расписании  $\pi$  упорядочим по правилу **EDD**.

Требование  $V_{2n+1}$  занимает или позицию  $(n + 1)$  или позицию  $n + 2$  при расписании  $\pi$ . Если требование  $V_{2n+1}$  занимает позицию  $(n + 2)$  "слева" то позиции  $n$  и  $(n + 1)$  "слева" занимают требования  $V_{2n-1}$  и  $V_{2n}$ , соответственно, согласно **циклу 1** и **EDD** сортировке.

Рассмотрим возможные ситуации.

I. Пусть требование  $V_{2n+1}$  занимает позицию  $n + 2$ .

Рассматривается расписание следующего вида  $\pi = (\pi_1, V_{2n-1}, V_{2n}, V_{2n+1}, \pi_2)$ , при котором требование  $V_{2n}$  обслуживается  $(n + 1)$  по порядку.

При частичных расписаниях  $\pi_1$  и  $\pi_2$  обслуживается по  $(n - 1)$  требованию, т.е.  $|\{\pi_1\}| = n - 1 = |\{\pi_2\}|$ .

Учитывая, что **циклом 1** не исключаются только ситуации описанные в лемме 26. Поэтому выполняется  $P(\pi_1) + 2qb + \delta > P(\pi_2) > P(\pi_1) + 2qb - \delta$ , где  $q$  - количество ситуаций описанных в лемме 26 при расписании  $\pi$ .

**Примером** рассматриваемой ситуации может служить расписание, при котором множества:

$$\{\pi_1\} = \{V_{2i-1}, V_{2i}\} \cup \{V_{1,1}, V_{2,1}, \dots, V_{i-2,1}, V_{i+1,1}, \dots, V_{n-1,1}\};$$

$$\{\pi_2\} = \{V_{2(i-1)-1}, V_{2(i-1)}\} \cup \{V_{1,2}, V_{2,2}, \dots, V_{i-2,2}, V_{i+1,2}, \dots, V_{n-1,2}\}.$$

Тогда  $q = 1$  и  $P(\pi_1) + 2b + \delta > P(\pi_2) > P(\pi_1) + 2b - \delta$ , так как

$$\begin{aligned}
& -(\delta - \delta_{i-1} - \delta_i - \delta_n) < \\
& P(\{V_{1,1}, V_{2,1}, \dots, V_{i-2,1}, V_{i+1,1}, \dots, V_{n-1,1}\}) - \\
& P(\{V_{1,2}, V_{2,2}, \dots, V_{i-2,2}, V_{i+1,2}, \dots, V_{n-1,2}\}) < \delta - \delta_{i-1} - \\
& \delta_i - \delta_n \text{ и}
\end{aligned}$$

$$P(\{V_{2(i-1)-1}, V_{2(i-1)}\}) - P(\{V_{2i-1}, V_{2i}\}) = 2b + \delta_{i-1} - \delta_i.$$

Рассмотрим две ситуации, когда  $q = 1$  и  $q > 1$ . При  $q = 0$  расписание  $\pi$  имеет вид (10) (см. лемму 23). Этот случай мы рассмотрели выше.

а) Пусть  $q = 1$ .

$$\text{Известно, что } \sum_{i=1}^{2n+1} p_i = 2L + p_{2n+1} = 2L + n^3b.$$

Обозначим через  $\Delta = P(\pi_2) - (P(\pi_1) + 2b)$ , для которой верно  $-\delta < \Delta < \delta$ .

$$\text{Пусть } S = P(\pi_1). \text{ Тогда } 2S + 2b + \Delta + p_{2n-1} + p_{2n} + p_{2n+1} = 2S + \Delta + 2b + 3n^3b + 2b + \delta_n = 2L + n^3b.$$

Поэтому

$$L = S + \frac{1}{2}\Delta + 2b + n^3b + \frac{1}{2}\delta_n,$$

тогда

$$\begin{aligned}
c_{2n}(\pi) &= P(\pi_1) + p_{2n-1} + p_{2n} = S + 2n^3b + 2b + \delta_n = \\
&= L + n^3b + \frac{1}{2}\delta_n - \frac{1}{2}\Delta.
\end{aligned}$$

Известно, что  $L + n^3b = d_{2n+1}$ , тогда выполняется  $-\delta < c_{2n}(\pi) - d_{2n+1} < \delta$ .

Необходимо рассмотреть два подслучая, когда  $c_{2n}(\pi) \geq d_{2n+1}$  и  $c_{2n}(\pi) < d_{2n+1}$ .

$$1. c_{2n}(\pi) \geq d_{2n+1}.$$

Рассмотрим расписание  $\pi'$  следующего вида  
 $\pi' = (\pi_1, V_{2n-1}, V_{2n+1}, V_{2n}, \pi_2).$

$$\begin{aligned} F(\pi) - F(\pi') &= T_{2n}(\pi) + T_{2n+1}(\pi) - (T_{2n}(\pi') + \\ &+ T_{2n+1}(\pi')) = (T_{2n+1}(\pi) - T_{2n+1}(\pi')) - (T_{2n}(\pi') - \\ &- T_{2n}(\pi)) = (p_{2n+1} + (c_{2n}(\pi) - d_{2n+1})) - p_{2n+1} = \\ &c_{2n}(\pi) - d_{2n+1} \geq 0. \end{aligned}$$

$$2. c_{2n}(\pi) < d_{2n+1}.$$

При этом  $c_{2n}(\pi) > d_{2n}$ , так как  $d_{2n+1} - d_{2n} = \delta$   
и  $d_{2n+1} - c_{2n}(\pi) < \delta$ .

Распишем структуру расписания  $\pi$ .

$$\pi = (\pi_{11}, V_{2i-1}, V_{2i}, \pi_{12}, V_{2n-1}, V_{2n}, V_{2n+1}, \pi_{21}, X, \pi_{22}),$$

где  $|\{\pi_{22}\}| = i - 1$ ,  $X \in \{V_{2(i-1)-1}, V_{2(i-1)}\}$ .

Если  $X = V_{2(i-1)-1}$ , то транспозиция соседних требований  $V_{2(i-1)-1}$  и  $V_{2(i-1)}$  согласно правилу SPT не увеличит значение целевой функции.

Пусть  $X = V_{2(i-1)}$ . При расписании  $\pi$  запаздывает  $(n + 1)$  требование.

Построим расписание

$$\pi' = (\pi_{11}, V_{2i-1}, X, \pi_{12}, V_{2n-1}, V_{2n}, V_{2n+1}, \pi_{21}, V_{2i}, \pi_{22}).$$

Выполняется

$$\begin{aligned} F(\pi) - F(\pi') &= (d_{2i} - d_{2(i-1)}) - (n - i + 1)(p_{2(i-1)} - \\ &- p_{2i}) = (n - i + 1)b - \delta - (n - i + 1)b = -\delta, \text{ т.е.} \\ &\text{значение целевой функции увеличилось на } \delta. \end{aligned}$$

Тогда  $c_{2n}(\pi') - d_{2n+1} > b - \delta$ . Построим расписание

$$\pi'' = (\pi_{11}, V_{2i-1}, X, \pi_{12}, V_{2n-1}, V_{2n+1}, V_{2n}, \pi_{21}, V_{2i}, \pi_{22}).$$

Оценим разность значений целевой функции  $F(\pi') - F(\pi'') > (p_{2n+1} + b - \delta) - p_{2n+1} > b - \delta$ .

Тогда  $F(\pi) - F(\pi'') = b - \delta - \delta > 0$ .

б) Пусть  $q > 1$ . Тогда  $d_{2n} - c_{2n}(\pi) > b - 2\delta$ .

Если  $q = 2$ , то при расписании  $\pi'$ , рассмотренном в лемме 26, для требования  $Y = V_{2n}$  выполняется  $T_Y(\pi') < 2\delta$ . Перестановка, описанная в лемме 26, уменьшит значение целевой функции.

Если  $q > 2$ , то при расписании  $\pi'$  будет запаздывать  $n$  требований, поэтому, согласно лемме 26, имеет место неравенство  $F(\pi) > F(\pi')$ .

II. Пусть требование  $V_{2n+1}$  обслуживается  $(n + 1)$  по порядку. То в ситуации, описанной в лемме 26, будем иметь  $T_Y(\pi') = T_{2n+1}(\pi') < \frac{1}{2}\delta$ . Преобразование расписания, согласно лемме 26, уменьшит значение целевой функции.

**Цикл 2.** Пока среди незапаздывающих требований при очередном расписании  $\pi$  присутствует пара требований  $V_{2i-1}, V_{2i}$ , причем на позиции  $i$  "справа" обслуживается требование  $X \in \{V_{2(i-1)-1}, V_{2(i-1)}\}$ . Применяем для требований  $X$  и  $V_{2i}$  перестановку, описанную в пунктах I и II. При этом значение целевой функции уменьшается.

**Конец цикла 2.**

**Конец алгоритма преобразования исходного неканонического расписания.**

Таким образом, произвольное неканоническое расписание  $\pi$  за  $O(n)$  операций можно преобразовать к каноническому расписанию  $\pi^*$ . Причем  $F(\pi) > F(\pi^*)$ .  $\square$

**Теорема 10.** Решение примера **ЧНР** будет "ДА" тогда и только тогда, когда при оптимальном каноническом расписании  $c_{2n+1}(\pi) = d_{2n+1}$ .

**Доказательство.**

Рассмотрим каноническое расписание вида

$$\pi = (V_{1,1}, V_{2,1}, \dots, V_{i,1}, \dots, V_{n,1}, V_{2n+1}, V_{n,2}, \dots, V_{i,2}, \dots, V_{2,2}, V_{1,2})$$

Известно, что требования  $V_{n,2}, \dots, V_{i,2}, \dots, V_{2,2}, V_{1,2}$  запаздывают. Требование  $V_{2n+1}$  может запаздывать, тогда  $F(\pi) = \sum_{i=1}^n T_{V_{i,2}}(\pi) + T_{V_{2n+1}}(\pi)$ .

Обозначим через  $\sum_{i=1}^{2n+1} p_i = C$ .

Тогда

$$\sum_{i=1}^n c_{V_{i,2}}(\pi) = nC - \sum_{i=1}^{n-1} (n-i)p_{V_{i,2}}.$$

Обозначим

$$\phi(i) = \begin{cases} 1, & V_{i,2} = V_{2i-1}, \\ 0, & V_{i,2} = V_{2i}, \end{cases}$$

тогда

$$d_{V_{i,2}} = d_{2n+1} - \left( \sum_{k=i}^{n-1} (n-k)b + (n-i+1)\delta + \phi(i)((n-i)\delta_i + \varepsilon\delta_i) \right),$$

поэтому

$$\begin{aligned} \sum_{i=1}^n T_{V_{i,2}}(\pi) &= nC - \sum_{i=1}^{n-1} (n-i)p_{V_{i,2}} - \\ &- \sum_{i=1}^n (d_{2n+1} - \left( \sum_{k=i}^{n-1} (n-k)b + (n-i+1)\delta + \phi(i)((n-i)\delta_i + \varepsilon\delta_i) \right)). \end{aligned}$$

Задача  $\min_{\pi} F(\pi) = \min(\sum_{i=1}^n T_{V_{i,2}}(\pi) + T_{V_{2n+1}}(\pi))$  сводится к задаче максимизации функции  $\Phi$ , где функция  $\Phi = \sum_{i=1}^{n-1} (n-i)p_{V_{i,2}} - \sum_{i=1}^n \phi(i)((n-i)\delta_i + \varepsilon\delta_i) - T_{V_{2n+1}}(\pi)$ .

1. Если  $V_{i,2} = V_{2i}$ ,  $i = 1, \dots, n$ , то  $T_{V_{2n+1}}(\pi) = \frac{1}{2}\delta$ ,  
 $\Phi_1 = \sum_{i=1}^{n-1} (n-i)p_{2i} - \frac{1}{2}\delta$ .
2. Если  $V_{i,2} = V_{2i-1}$ ,  $i = 1, \dots, n$ , то  
 $T_{V_{2n+1}}(\pi) = \max\{-\frac{1}{2}\delta, 0\} = 0$ ,  
 $\Phi = \sum_{i=1}^{n-1} (n-i)p_{2i-1} - \sum_{i=1}^n ((n-i)\delta_i + \varepsilon\delta_i) =$   
 $\sum_{i=1}^{n-1} (n-i)p_{2i} + \sum_{i=1}^{n-1} (n-i)\delta_i - \sum_{i=1}^n ((n-i)\delta_i + \varepsilon\delta_i) =$   
 $\Phi_1 + \frac{1}{2}\delta - \sum_{i=1}^n \varepsilon\delta_i$ .

Функция  $\Phi$  достигает максимальное значение равное  $\Phi_1 + \frac{1}{2}\delta - \frac{1}{2} \sum_{i=1}^n \varepsilon\delta_i$ , когда  $\sum_{i=1}^n \phi(i)(\varepsilon\delta_i) = \frac{1}{2} \sum_{i=1}^n \varepsilon\delta_i$ , что равнозначно  $\sum_{i=1}^n \phi(i)\delta_i = \frac{1}{2} \sum_{i=1}^n \delta_i$ . Следовательно для модифицированного примера существуют два подмножества  $A_1$  и  $A_2$ , таких что  $\sum_{a_i \in A_1} a_i = \sum_{a_i \in A_2} a_i$  (ответ "ДА"). При этом  $c_{2n+1}(\pi) = d_{2n+1}$ .

Если решение модифицированного примера **ЧНР** "НЕТ", то не выполняется равенство  $\sum_{i=1}^n \phi(i)\delta_i = \frac{1}{2} \sum_{i=1}^n \delta_i$ . Учитывая значение  $d_{2n+1}$ , будем иметь  $c_{2n+1}(\pi) \neq d_{2n+1}$ .

Если  $c_{2n+1}(\pi) = d_{2n+1}$ , то из этого следует, что выполняется  $\sum_{i=1}^n p_{V_{i,1}} = \sum_{i=1}^n p_{V_{2i}} + \frac{1}{2}\delta = \sum_{i=1}^n p_{V_{i,2}}$ , т.е. решение модифицированного примера **ЧНР** будет иметь ответ "ДА".  $\square$

## 5. Трудоемкость известных алгоритмов

Будет показано, что трудоемкость известных алгоритмов [7, 11, 13] решения задачи  $1 \parallel \sum T_j$  не меньше  $O(n2^{(n-1)/3-1})$  для *канонических DL-примеров* и не меньше  $O(n2^{(n-1)/2})$  для примеров случая **ВФ**.

В основе известных алгоритмов [7, 8, 11, 13] используются правила сокращения перебора: **правила исключения 1-4**, **анализ параметров  $E_j$ ,  $L_j$** , **построение модифицированного примера**. Показано, что алгоритмы, использующие только эти правила сокращения перебора, имеют экспоненциальную трудоемкость (от количества требований  $n$ ) для *канонических* случаев проблемы  $1 \parallel \sum T_j$ .

В первом подразделе исследуется трудоемкость известных алгоритмов для *канонических примеров DL*. Предложен альтернативный алгоритм трудоемкости  $O(n\delta)$ .

Частный случай ВФ, для которого существует алгоритм трудоемкости  $O(n^2)$ , анализируется в разд. 3.

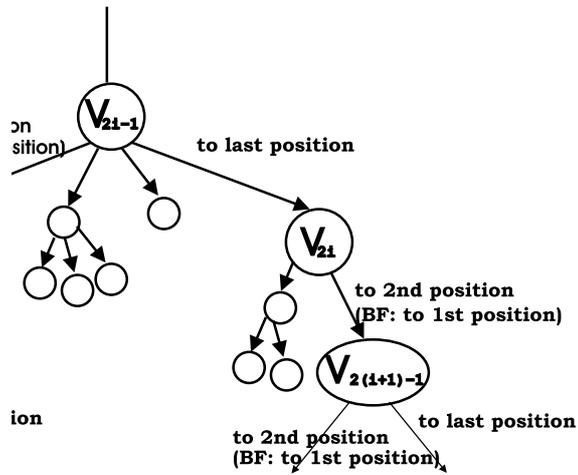


Рис. 5. Дерево поиска

### 5.1. Канонические DL-примеры задачи $1 || \sum T_j$

Далее будет показано, что алгоритмы поиска оптимального расписания, в которых используются известные методы сокращения перебора (Правила исключения 1-4, использование  $E_j$  и  $L_j$ , построение модифицированного примера) [7, 11], в случае канонических DL-примеров имеют экспоненциальную трудоемкость.

**Определение.** Канонические примеры, для которых выполняется

$$\frac{1}{2}\delta - \sum_{j=1}^{i-1} \delta_j \geq \delta_i, \quad 2 \leq i \leq (n-1),$$

будем называть примерами случая SD (short delta).

Для случая  $SD$  выполняется

$$\delta_i > \frac{\sum_{j:=1}^{i-1} \delta_j - \frac{1}{2}\delta}{2(n-i+1)}, \quad 2 \leq i \leq (n-1),$$

так как  $\frac{1}{2}\delta - \sum_{j:=1}^{i-1} \delta_j \geq \delta_i > 0$ , то  $\sum_{j:=1}^{i-1} \delta_j - \frac{1}{2}\delta < 0$ .

Например, случаю  $SD$  удовлетворяют примеры, для которых

$$\delta_i > 2 \sum_{j:=1}^{i-1} \delta_j, \quad 2 \leq i \leq n.$$

Требования  $N$  пронумеруем в порядке EDD (early due date):  $(V_1, V_2, W_1, \dots, V_{2i-1}, V_{2i}, W_i, \dots, W_n, W_{n+1})$ .

На рис. 5 представлено дерево поиска оптимального расписания **алгоритма А** для *канонических DL-примеров* с использованием перебора подходящих позиций.

**Определение.** *Остов дерева поиска, получаемый "двоичным ветвлением" (рис. 5) будем называть "основным деревом".*

**Лемма 27.** При использовании правил исключения 1–3 дерево поиска содержит "двоичное ветвление" (рис. 5). Ветвление происходит при выборе места для очередного требования  $V_{2i-1}$ . Для требования  $V_{2i}$  допустимой становится "противоположная" позиция.

**Доказательство.** Доказательство проведем методом математической индукции.

1. Покажем, что "двоичное ветвление" имеет место при выборе позиций для требований  $V_1$  и  $V_2$ .

Имеем множество  $N$ , состоящее из  $3n + 1$  требований,  
 $N = \{V_1, V_2, W_1, \dots, V_{2i-1}, V_{2i}, W_i, \dots, W_n, W_{n+1}\}$ .

Требование  $j^* = V_1$ , момент начала обслуживания требований  $t' = 0$ .

а. Покажем, что позиция 1 для требования  $V_1$  является подходящей.

Очевидно, что правило **б** выполняется.

Правило **а** также выполняется, так как

$$t' + p_{V_1} = 0 + a_1 < d_{V_2} = \frac{1}{2}\delta + a_2 + 2(n-1+1)(a_1 - a_2),$$

т.е. позиция 1 для требования  $V_1$  подходящая.

Покажем, что при постановке  $V_1$  на позицию 1 для требования  $V_2$  (которое на второй итерации является  $j^*$ ) позиция  $(3n+1)$  будет подходящей.

Правило **б** выполняется. Легко убедиться, что

$$t' + \sum_{j=1}^{3n} p_j = \sum_{j=1}^{2n} a_j + (n+1)b >$$

$$> (n+1)b + \frac{1}{2}\delta + a_2 + a_4 + \dots + a_{2n} + a_2.$$

$$(n+1)b + \frac{1}{2}\delta + a_2 + a_4 + \dots + a_{2n} + a_2 =$$

$$= d_{W_{n+1}} + a_2 = \max_{j \in N'} \{d_j\} + \max_{j \in N'} \{p_j\} > d_j + p_j, \forall j \in N'.$$

Правило **а**, очевидно, выполняется, так как позиция  $3n+1$  последняя.

б. Покажем, что позиция  $3n+1$  для требования  $V_1$  подходящая.

Правило **а**, очевидно, выполняется, так как позиция  $3n+1$  последняя.

Покажем выполнение правила **б**. Видно, что

$$\begin{aligned}
 0 + \sum_{j=1}^{3n+1} p_j &= \sum_{j=1}^{2n} a_j + (n+1)b > \\
 &> (n+1)b + \frac{1}{2}\delta + a_2 + a_4 + \dots + a_{2n} + a_1.
 \end{aligned}$$

$$\begin{aligned}
 (n+1)b + \frac{1}{2}\delta + a_2 + a_4 + \dots + a_{2n} + a_1 &= d_{W_{n+1}} + a_1 = \\
 &= \max_{j \in N} \{d_j\} + \max_{j \in N} \{p_j\} > d_j + p_j, \forall j \in N
 \end{aligned}$$

Покажем, что если требование  $V_1$ , обслуживается  $3n+1$  (последним) по порядку, то для требования  $V_2$  становится подходящей только позиция 1.

Очевидно, что правило **б** выполняется.

Правило **а** также выполняется, так как

$$t' + p_{V_2} = 0 + a_2 < d_{W_1} = b + a_2.$$

2. Предположим, что "двоичное ветвление" имело место в группах  $1, 2, \dots, i-1$ . В основном дереве первое требование из каждой пары занимает "вторую" (после соответствующего требования  $W$ ) или "последнюю" позицию, а второе требование из пары - "противоположную" позицию.
3. В предположении п. 2 имеет место неравенство:

$$a_2 + a_4 + \dots + a_{2i-2} + (i-2)b \leq t' \leq a_1 + a_3 + a_{2i-3} + (i-2)b,$$

Рис. 6. Текущие "вторая" и "последняя" позиции

где  $t'$  – момент времени, в котором "принимается решение" относительно  $i$ -ой группы требований  $\{W_{i-1}, V_{2i-1}, V_{2i}\}$ . Обозначим через  $\bar{\delta}$  величину  $\bar{\delta} = t' - (a_2 + a_4 + \dots + a_{2i-2} + (i-2)b)$ , для которого выполняется  $0 \leq \bar{\delta} \leq \sum_{j=1}^{i-1} \delta_j < \delta$ . Тогда

$$t' = a_2 + a_4 + \dots + a_{2i-2} + (i-2)b + \bar{\delta}.$$

Имеется множество требований

$N' = \{W_{i-1}, V_{2i-1}, V_{2i}, W_i, \dots, W_n, W_{n+1}\}$ , для которого  $j^* = V_{2i-1}$ .

Учитывая, что  $d_{W_{i-1}} < d_j$ ,  $p_{W_{i-1}} \leq p_j$ ,  $\forall j \in N' \setminus \{W_{i-1}\}$ , то при любом оптимальном расписании  $\pi$  выполняется  $(W_{i-1} \rightarrow j)_\pi$ ,  $\forall j \in N' \setminus \{W_{i-1}\}$ .

"Вторая" и "последняя" позиции для требования  $V_{2i-1}$  являются подходящими:

а. покажем, что текущая "вторая" позиция для требования  $V_{2i-1}$  подходящая.

Правило **б** выполняется, так как  $t' + p_{W_{i-1}} + p_{V_{2i-1}} = t' + b + a_{2i-1} = a_2 + a_4 + \dots + a_{2i-2} + (i-2)b + \bar{\delta} + a_{2i-1} + b > d_{W_{i-1}} + p_{W_{i-1}} = a_2 + a_4 + \dots + a_{2i-2} + (i-2)b + b$ .

Покажем, что выполняется и правило **а**.

Для случая  $SD$  выполняется:  $t' + b + a_{2i-1} = a_2 + a_4 + \dots + a_{2i-2} + (i-1)b + \bar{\delta} + a_{2i-1} < (i-1)b + \frac{1}{2}\bar{\delta} + (a_2 + a_4 + \dots + a_{2i-2} + a_{2i}) + 2(n-i+1)(a_{2i-1} - a_{2i}) = d_{V_{2i}}$ . Покажем, что при постановке  $V_{2i-1}$  во "вторую" (после  $W_{i-1}$ ) позицию для требования  $V_{2i}$  становится подходящей "последняя" позиция в списке  $N'' = \{V_{2i}, W_i, \dots, W_n, W_{n+1}\}$ . Правило **б** также выполняется:

$$t' + b + a_{2i-1} + \sum_{j \in N''} p_j = a_2 + a_4 + \dots + a_{2i-2} + (i-1)b + \bar{\delta} + (n-i+2)b + \sum_{j=2i-1}^{2n} a_j > a_2 + a_4 + \dots + a_{2i-2} + a_{2i} + a_{2i+2} + \dots + a_{2n} + (n+1)b + a_{2i},$$

поэтому

$$a_2 + a_4 + \dots + a_{2i-2} + a_{2i} + a_{2i+2} + \dots + a_{2n} + (n+1)b + a_{2i} = d_{W_{n+1}} + a_{2i} = \max_{j \in N''} d_j + \max_{j \in N''} p_j > d_j + p_j, \forall j \in N''.$$

Правило **а** для "последней" позиции, очевидно, выполняется, т. к. директивный срок "фиктивного требования"  $d_{|N''|+1} = +\infty > t' + b + a_{2i-1} + \sum_{j \in N''} p_j$ .

б. Покажем, что текущая "последняя" позиция для требования  $V_{2i-1}$  является также подходящей.

Правило **а**, очевидно, выполняется, т. к. имеем  $d_{|N'|+1} = +\infty > t' + \sum_{j \in N'} p_j$ .

Покажем выполнение правила **б**, т.к.

$$t' + \sum_{j \in N'} p_j = a_2 + a_4 + \dots + a_{2i-2} + (i-1)b + \bar{\delta} + (n-i+2)b + \sum_{j=2i}^{2n} a_j > a_2 + a_4 + \dots + a_{2i-2} + a_{2i} + a_{2i+2} + \dots + a_{2n} + (n+1)b + a_{2i-1},$$

поэтому

$$a_2 + a_4 + \dots + a_{2i-2} + a_{2i} + a_{2i+2} + \dots + a_{2n} + (n+1)b + a_{2i-1} = d_{W_{n+1}} + a_{2i-1} = \max_{j \in N'} d_j + \max_{j \in N'} p_j > d_j + p_j, \forall j \in N',$$

т.е. правило **б** выполняется.

Покажем, что при постановке  $V_{2i-1}$  в "последнюю" позицию для требования  $V_{2i}$  становится подходящей "вторая" позиция в списке  $N'' = \{W_{i-1}, V_{2i}, W_i, \dots, W_n, W_{n+1}\}$ .

Правило **б** выполняется, т.к.  $t' + b + a_{2i} = a_2 + a_4 + \dots + a_{2i-2} + (i-2)b + \bar{\delta} + b + a_{2i} > d_{W_{i-1}} + p_{W_{i-1}} = a_2 + a_4 + \dots + a_{2i-2} + (i-2)b + b$ .

Покажем, что выполняется и правило **а**:

$t' + b + a_{2i} = a_2 + a_4 + \dots + a_{2i-2} + (i-1)b + \bar{\delta} + a_{2i} < < ib + (a_2 + a_4 + \dots + a_{2i-2} + a_{2i}) = d_{W_i}$ , т. к. верно  $b \gg \delta > \bar{\delta}$ .  $\square$

**Лемма 28.** Правило исключения 4 не сокращает "двоичное ветвление" при выборе позиции для очередного требования  $V_{2i-1}$  в случае  $SD$ .

**Доказательство.** Доказательство проводится аналогично доказательству леммы 27.

Пусть "двоичное ветвление" имело место начиная с первой группы до группы  $i-1$ . Тогда

$$t' = a_2 + a_4 + \dots + a_{2i-2} + (i-2)b + \bar{\delta}, \quad 0 \leq \bar{\delta} \leq \sum_{j=1}^{i-1} \delta_j < \delta.$$

1. Покажем, что Правило исключения 4 не исключает из списка подходящих позиций для требования  $V_{2i-1}$  "вторую" позицию в списке  $N' = \{W_{i-1}, V_{2i-1}, V_{2i}, W_i, \dots, W_n, W_{n+1}\}$ . (Первую позицию займет требование  $W_{i-1}$ ).

Пусть  $\pi_1 = (W_{i-1}, V_{2i-1}, V_{2i}, W_i, \dots, W_n, W_{n+1})$ ,

$\pi_2 = (W_{i-1}, V_{2i}, V_{2i-1}, W_i, \dots, W_n, W_{n+1})$ .

Покажем, что  $F(\pi_2) - F(\pi_1) > 0$ .

$$\begin{aligned} F(\pi_2) - F(\pi_1) &= T_{V_{2i}}(\pi) + T_{V_{2i-1}}(\pi) - T_{V_{2i-1}}(\pi') + T_{V_{2i}}(\pi') = \\ &= \max\{t' + b + a_{2i} - d_{a_{2i}}, 0\} + \max\{t' + b + a_{2i} + a_{2i-1} - \end{aligned}$$

$$-d_{V_{2i-1}}, 0\} - \max\{t' + b + a_{2i-1} - d_{V_{2i-1}}, 0\} - \\ - \max\{t' + b + a_{2i-1} + a_{2i} - d_{a_{2i}}, 0\}.$$

Очевидно, что  $\max\{t' + b + a_{2i} + a_{2i-1} - d_{V_{2i-1}}, 0\} - \\ - \max\{t' + b + a_{2i-1} + a_{2i} - d_{V_{2i}}, 0\} = 2(n-i+1)(a_{2i-1} - a_{2i})$   
(из условия канонического DL-примера).

а. Пусть  $t' + b + a_{2i} - d_{V_{2i}} > 0$ , то  $t' + b + a_{2i-1} - d_{V_{2i-1}} > 0$   
(из условия канонического DL-примера).

Тогда  $F(\pi_2) - F(\pi_1) = 2(n-i+1)(a_{2i-1} - a_{2i}) + (a_{2i} - \\ - a_{2i-1} - 2(n-i+1)(a_{2i-1} - a_{2i})) < 0$ . Правило  
исключения 4 сокращает список "подходящих"  
позиций, но при этом выполняется

$$t' + b + a_{2i} - d_{V_{2i}} > 0 \Rightarrow a_2 + a_4 + \dots + a_{2i-2} + a_{2i} + \\ + (i-1)b + \bar{\delta} > (i-1)b + (a_2 + a_4 + \dots + a_{2i-2} + a_{2i}) + \\ + \frac{1}{2}\delta + 2(n-i+1)(a_{2i-1} - a_{2i}), \text{ поэтому верно } \bar{\delta} > \\ \frac{1}{2}\delta + 2(n-i+1)(a_{2i-1} - a_{2i}). \text{ Для случая } SD \text{ это не} \\ \text{выполняется, так как } \bar{\delta} \leq \sum_{j=1}^{i-1} \delta_j < \frac{1}{2}\delta.$$

б.  $t' + b + a_{2i} - d_{V_{2i}} \leq 0$  и  $t' + b + a_{2i-1} - d_{V_{2i-1}} > 0$ .

$$t' + b + a_{2i} - d_{V_{2i}} \leq 0 \Rightarrow a_2 + a_4 + \dots + a_{2i-2} + a_{2i} + (i-1)b + \bar{\delta} \leq (i-1)b + (a_2 + a_4 + \dots + a_{2i-2} + a_{2i}) + \frac{1}{2}\delta + 2(n-i+1)(a_{2i-1} - a_{2i}) \Rightarrow \bar{\delta} \leq \frac{1}{2}\delta + 2(n-i+1)(a_{2i-1} - a_{2i}).$$

$$F(\pi_2) - F(\pi_1) = 2(n-i+1)(a_{2i-1} - a_{2i}) - (\bar{\delta} - \frac{1}{2}\delta + \\ + (a_{2i-1} - a_{2i})) \geq 2(n-i+1)(a_{2i-1} - a_{2i}) - (\frac{1}{2}\delta + 2(n-i+1)(a_{2i-1} - a_{2i}) - \frac{1}{2}\delta + (a_{2i-1} - a_{2i})) = -(a_{2i-1} - a_{2i}) < 0. \\ \text{То есть Правило исключения 4 сокращает список} \\ \text{"подходящих" позиций, но имеет место:}$$

$$t' + b + a_{2i-1} - d_{V_{2i-1}} > 0 \Rightarrow a_2 + a_4 + \dots + a_{2i-2} + \\ + a_{2i-1} + (i-1)b + \bar{\delta} > (i-1)b + (a_2 + a_4 + \dots + a_{2i-2} + \\ + a_{2i-1}) + \frac{1}{2}\delta \Rightarrow \bar{\delta} > \frac{1}{2}\delta + (a_{2i} - a_{2i-1}). \text{ Для случая } SD \\ \text{это не выполняется, так как } \bar{\delta} \leq \sum_{j=1}^{i-1} \delta_j < \frac{1}{2}\delta.$$

с. Пусть  $t' + a_{2i} - d_{V_{2i}} < 0$  и  $t' + a_{2i-1} - d_{V_{2i-1}} < 0$ .  
 $F(\pi_2) - F(\pi_1) = 2(n - i + 1)(a_{2i-1} - a_{2i}) > 0$ .

2. Тогда Правило исключения 4 не исключает из списка подходящих позиций для требования  $V_{2i-1}$  "последнюю" позицию для требований подмножества  $N' = \{W_{i-1}, V_{2i-1}, V_{2i}, W_i, \dots, W_n, W_{n+1}\}$ .

Рассмотрим частичное расписание требований множества  $N'$   $\pi = (W_{i-1}, V_{2i}, W_i, \dots, W_n, W_{n+1}, V_{2i-1})$ , обслуживание которого начинается с момента времени  $t'$ . Очевидно, что  $d_{V_{2i}} \leq t' + p_{W_{i-1}} + p_{V_{2i}}$ ,  $d_{W_i} \leq t' + p_{W_{i-1}} + p_{V_{2i}} + p_{W_i}$ ,  $d_{W_{i-1}} \leq t' + p_{W_{i-1}}$ . Все остальные требования из  $\pi$  запаздывают (см. условие канонического *DL-примера*).

Обозначим расписание  $\pi = (W_{i-1}, \pi_1, \pi_2, V_{2i-1})$ . Рассмотрим расписание  $\pi' = (W_{i-1}, \pi_1, V_{2i-1}, \pi_2)$ . Очевидно, что в расписаниях  $\pi$  и  $\pi'$  требования из  $\pi_2$  и требование  $V_{2i-1}$  запаздывают.

При расписании  $\pi'$  суммарное запаздывание требований из  $\pi_2$  увеличилось на  $|\{\pi_2\}|a_{2i-1}$ , а запаздывание требования  $V_{2i-1}$  сократилось на  $P(\pi_2)$ .

$F(\pi') - F(\pi) = |\{\pi_2\}|a_{2i-1} - P(\pi_2) > 0$ , т. к. выполняется  $a_{2i-1} > p_j, j \in \{\pi_2\}$ .

Таким образом  $F(\pi) < F(j^*, i)$ ,  $j^* \leq i \leq k$ .

□

**Лемма 29.** Правило исключения 4 удаляет из списка "подходящих" позиций для очередного требования  $V_{2i-1}$ ,  $i = 1, 2, \dots, n$ , все позиции кроме текущих "второй" и "последней".

**Доказательство.** Пусть

$$t' = a_2 + a_4 + \dots + a_{2i-2} + (i-2)b + \bar{\delta}, \quad 0 \leq \bar{\delta} \leq \sum_{j=1}^{i-1} \delta_j < \delta.$$

Имеем множество требований

$$N' = \{W_{i-1}, V_{2i-1}, V_{2i}, W_i, \dots, W_n, W_{n+1}\}.$$

Покажем, что  $F(V_{2i-1}, k) > F(V_{2i-1}, k+1)$ , где  $k$  - не "вторая" и не "последняя" позиция.

Рассмотрим 2 расписания, построенных с момента времени  $t'$ :

$$\begin{aligned} \pi_1 &= (W_{i-1}, V_{2i}, W_i, V_{2(i+1)-1}, V_{2(i+1)}, \dots, V_{2i-1}, X, \dots, W_{n+1}) \text{ и} \\ \pi_2 &= (W_{i-1}, V_{2i}, W_i, V_{2(i+1)-1}, V_{2(i+1)}, \dots, X, V_{2i-1}, \dots, W_{n+1}). \end{aligned}$$

Не трудно показать, что все требования из  $\pi_1$  и  $\pi_2$  запаздывают (кроме, быть может,  $W_{i-1}$ ,  $V_{2i}$  и  $W_i$ ).

Тогда  $F(\pi_1) > F(\pi_2)$ , так как  $p_{V_{2i-1}} > p_X$  для любого требования  $X \in N' \setminus V_{2i-1}$ .  $\square$

Аналогичные рассуждения можно провести для требования  $V_{2i}$ . Следовательно, при помощи алгоритма, использующего Правило исключения 4, строятся только канонические расписания.

**Лемма 30.** При использовании значений  $E_j, L_j$  для примеров случая  $SD$  "двоичное ветвление" в дереве поиска не сокращается.

**Доказательство.** Для списка требований  $N$ , при использовании правила Эммонса выполняется только  $(W_{i-1} \rightarrow j)_\pi, \forall j \in \{V_{2i-1}, V_{2i}, W_i, \dots, W_{n+1}\}$  при любом каноническом DL-расписании. Тогда  $E_{V_{2i-1}} = (i-1)b + a_{2i-1}, E_{V_{2i}} = (i-1)b + a_{2i}, E_{W_i} = ib = d_{W_i}. \square$

**Лемма 31.** Для модифицированного примера двоичное ветвление в "основном дереве" сохранится.

**Доказательство.** Из леммы 30 и вида *канонического DL-расписания*

$$E_{V_{2i-1}} = (i-1)b + a_{2i-1} < d_{V_{2i-1}};$$

$$E_{V_{2i}} = (i-1)b + a_{2i} < d_{V_{2i}}; E_{W_i} = ib < d_{W_i},$$

т.е.  $d'_i = d_i, \forall i$ . Модифицированный пример не отличается от исходного, так как  $p'_i = p_i, d'_i = \max\{E_i, d_i\} = d_i, \forall i \in N. \square$

Величины  $b_i$  не влияют на ветвления основного дерева, влияют только  $\delta_i, i = 1, 2, \dots, n$ . Американские ученые Ду и Леунг (J. Du & J. Y.-T. Leung) показали, что если ответ соответствующего примера **ЧНР** "ДА", то количество оптимальных расписаний четно [5].

## 5.2. Трудоемкость известных алгоритмов для канонических DL-примеров

**Теорема 11.** Для случая канонических  $SD$ -примеров алгоритмы, использующие только следующие способы сокращения перебора: Правила исключения 1-4, использование  $E_j$  и  $L_j$ , построение модифицированного примера, имеют трудоемкость не меньше  $O(n2^{(n-1)/3-1})$  операций.

**Доказательство.** В лемме 27 показано, что для очередного требования  $V_{2i-1}$ ,  $i = 1, 2, \dots, n-1$  "подходящими" являются две позиции.

Правило исключения 4 не сокращает этот список из двух позиций (лемма 28). В этом случае рассматривается две подзадачи (когда требование  $V_{2i-1}$  занимает текущую "вторую" или текущую "последнюю" позицию).

Процедура поиска подходящих позиций для очередного требования с максимальной продолжительностью обслуживания в каждой подзадаче имеет трудоемкость  $O(n)$  операций.

Двоичное ветвление (рис. 5) выполняется для каждого требования  $V_{2i-1}$ ,  $i = 1, 2, \dots, n-1$ , количество которых  $(n-1)/3 - 1$ . Следовательно трудоемкость алгоритмов составляет не меньше  $O(n2^{(n-1)/3-1})$  операций.  $\square$

Заметим, если пример не удовлетворяет случаю  $SD$ , то "основное дерево" не будет полным. Ветвления в дереве поиска будут продолжаться до тех пор пока выполняется неравенство  $\bar{\delta} < \frac{1}{2}\delta + 2(n-i+1)(a_{2i-1} - a_{2i})$ . Если же величина разности  $a_{2k-1} - a_{2k} \approx a_{2l-1} - a_{2l}$ , когда индексы  $k, l = 1, \dots, n, k \neq l$ , то сохранится, по крайней мере, "половина дерева" (ветвление имеет место для требований  $i = 1, \dots, n/2$ ). Таким образом экспоненциальная трудоемкость

алгоритма сохранится.

Стоит отметить, что для модифицированного примера **ЧНР**, для которого имеет место  $\delta_1 \geq \dots \geq \delta_n$ , трудоемкость алгоритма, использующего правила исключения 1-4, будет наименьшей.

### 5.3. Трудоемкость известных алгоритмов для случая (2)

Примеры, удовлетворяющие следующим ограничениям, будем называть примерами случая  $BF2$ :

$$\left\{ \begin{array}{l} p_1 \geq p_2 \geq \dots \geq p_n, \\ d_1 \leq d_2 \leq \dots \leq d_n, \\ d_n - d_1 \leq p_n, \\ n = 2k, \\ \sum_{i=1}^k p_i < d_j < \sum_{i=k}^n p_i, \quad j = 1, 2, \dots, n, \\ p_1 - p_n \ll p_n, \\ \sum_{i=k+j+1}^n (p_{2j-1} - p_i) > d_{k+j} - d_{2j}, \quad j = 1, \dots, (k-1), \\ \sum_{i=k+j+1}^n (p_{2j} - p_i) > d_{k+j} - d_{2j}, \quad j = 1, \dots, (k-1). \end{array} \right. \quad (11)$$

Обозначим  $(1, 2, \dots, n) = (V_1, V_2, \dots, V_{2j-1}, V_{2j}, \dots, V_n)$ .

Не трудно убедиться, что при любом расписании для любого примера случая (11) запаздывать будет ровно  $k$  требований.

**Лемма 32.** Для примеров случая (11) дерево поиска содержит "двоичные ветвления" (рис.5). Ветвление происходит при выборе места для очередного требования  $V_{2i-1}$ . Для требования  $V_{2i}$  допустимой становится "противоположная" позиция,  $i = 1, \dots, (k-1)$ . Правила исключения 1-4 не сокращают "двоичные ветвления".

**Доказательство.** Доказательство проводится аналогично доказательству леммы 27 и леммы 28.

Предположим, что двоичное ветвление (когда первое требование из пары занимает "первую" или "последнюю" позицию, а второе требование из пары - "противоположную" позицию) имело место в группах  $1, 2, \dots, i-1$ ,  $i < k$ .

Рассматривается множество требований  $N' = \{V_{2i-1}, V_{2i}, \dots, V_n\}$ .

Множество требований  $N'$  начинает обслуживаться с момента времени  $t' = p_2 + p_4 + \dots + p_{2i-2} + \bar{\delta}$ , где  $0 \leq \bar{\delta} \leq \sum_{j=1}^{i-1} (p_{2j-1} - p_{2j}) < p_n$ .

Требование с максимальной продолжительностью  $j^* = V_{2i-1}$ . Покажем, что "первая" и "последняя" позиции для требования  $V_{2i-1}$  подходящие.

- а. Текущая "первая" позиция является для требования  $V_{2i-1}$  подходящей.

Очевидно, что правило **б** выполняется.

Покажем, что выполняется правило **а**:

$t' + p_{2i-1} = p_2 + p_4 + \dots + p_{2i-2} + \bar{\delta} + p_{2i-1} < d_{2i}$ .  
Требование  $V_{2i-1}$  при постановке на "первую" позицию не запаздывает. Можно показать, что не запаздывает и "следующее" требование  $V_{2i}$ .

Покажем, что Правило исключения 4 не исключает "первую" позицию. Рассмотрим 2 расписания  $\pi' = (\pi_1, V_{2i-1}, V_{2i}, \pi_2)$  и  $\pi'' = (\pi_1, V_{2i}, V_{2i-1}, \pi_2)$ , где  $P(\pi_1) = t'$ , легко убедиться, что  $F(\pi') = F(\pi'')$ , т. к. при обоих расписаниях требования  $V_{2i-1}$  и  $V_{2i}$  не запаздывают.

Проводя аналогичные рассуждения не трудно доказать, что при постановке требования  $V_{2i-1}$  в "последнюю" позицию, для требования  $V_{2i}$  допустимой становится "первая" позиция.

- б. Покажем, что текущая "последняя" позиция для требования  $V_{2i-1}$  подходящая.

Очевидно, что правило **а** выполняется, т. к.  $d_{|N'|+1} = +\infty > t' + \sum_{j \in N'} p_j$ .

Покажем выполнение правила **б**.

$$t' + \sum_{j \in N'} p_j = p_2 + p_4 + \dots + p_{2i-2} + \bar{\delta} + \sum_{j:=2i-1}^{2n} p_j > \sum_{j:=k}^n p_j + p_{2i-1}.$$

$$\sum_{i:=k}^n p_i + p_{2i-1} > \max_{j \in N'} d_j + \max_{j \in N'} p_j > d_j + p_j, \forall j \in N'.$$

Покажем, что правило исключения 4 не сокращает перебор.

Рассмотрим два частичных модифицированных EDD расписания  $\pi' = (\pi_1, \pi_2, V_{2i-1})$  и  $\pi'' = (\pi_1, V_{2i-1}, \pi_2)$ , составленные из требований множества  $N'$ , обслуживание которых начинается с момента времени  $t'$ .

Рассмотрим 2 случая.

1. Пусть при расписании  $\pi''$  требование  $V_{2i-1}$  запаздывает. Очевидно, что при расписаниях  $\pi'$  и  $\pi''$  запаздывают одни и те же требования. Для случая (2) запаздывающие требования при оптимальном расписании упорядочены в порядке SPT. Тогда  $F(\pi') < F(\pi'')$ .
2. Пусть при расписании  $\pi''$  требование  $V_{2i-1}$  не запаздывает. Представим расписания в виде:  $\pi' = (\pi_1, \pi_{21}, \pi_{22}, V_{2i-1})$  и  $\pi'' = (\pi_1, V_{2i-1}, \pi_{21}, \pi_{22})$ . Требования множества  $\{\pi_{22}\}$  запаздывают при обоих расписаниях. Требования множества  $\{\pi_{21}\}$ ,  $|\pi_{21}| > 0$ , при расписании  $\pi'$  не запаздывают. Учитывая, что количество запаздывающих требований при любом полном расписании равно  $n/2 = k$ , то не трудно вычислить  $|\{\pi_{22}\}|$ .

К шагу  $i$  уже выбрано  $i - 1$  запаздывающих требований, расположенных "в конце" полного расписания. Тогда  $|\{\pi_{22}\}| = k - (i - 1) - 1$ .

Множество  $\{\pi_{22}\} = \{V_{n-|\{\pi_{22}\}|+1}, \dots, V_n\} = \{V_{k+i+1}, \dots, V_n\}$ . Тогда при расписании  $\pi''$  стало запаздывать еще одно требование  $V_{k+i} \in \{\pi_{21}\}$ .

$$F(\pi') - F(\pi'') = c_{2i-1} - d_{2i-1} - |\{\pi_{22}\}|p_{2i-1} - (c_{k+i} - d_{k+i}) = t' + \sum_{i \in \{\pi_1\} \cup \{\pi_{21}\}} p_i + \sum_{i \in \{\pi_{22}\}} p_i + p_{2i-1} - d_{2i-1} - |\{\pi_{22}\}|p_{2i-1} - (t' + \sum_{i \in \{\pi_1\} \cup \{\pi_{21}\}} p_i + p_{2i-1} - d_{k+i}) = d_{k+i} - d_{2i-1} - \sum_{j:k+i+1}^n (p_{2i-1} - p_j).$$

Для случая (11) выполняется  $d_{k+i} - d_{2i-1} < \sum_{j:k+i+1}^n (p_{2i-1} - p_j)$ , то  $F(\pi') - F(\pi'') < 0$ . Тогда  $F(V_{2i-1}, i) > F(V_{2i-1}, |N'|)$ ,  $1 \leq i < |N'|$ . Правило исключения 4 не сокращает перебор.

Аналогичные рассуждения можно провести для "последней" позиции требования  $V_{2i}$ , когда требование  $V_{2i-1}$  поставлено на "первую" позицию.  $\square$

Нетрудно показать, что  $E_j = p_j$ ,  $L_j = \sum_{i=1}^n p_i$ . Тогда модифицированный пример, для которого  $p'_j = p_j$ ,  $d'_j = \max\{E_j = p_j, d_j\} = d_j$  совпадает с исходным примером. Значения  $E_j$ ,  $L_j$  не сокращают "двоичное ветвление".

**Теорема 12.** Для случая (11) алгоритмы, использующие только следующие правила исключения: правила исключения 1-4, использование  $E_j$  и  $L_j$ , построение модифицированного примера, имеют трудоемкость  $O(n2^{n/2})$ .

**Доказательство.**

Аналогично доказательству теоремы 11.  $\square$

Необходимо отметить, что для случая (11) **Алгоритм ВФ** находит оптимальное расписания за время  $O(n^2)$ .

## 6. Свойства случая В-1

Пусть  $d_j(t) = d_j - d_n + t$ ,  $j \in N$ , где  $t \in R$ .

Определим функцию  $F(\pi, t) = \sum_{j=1}^n \max\{0, c_j(\pi) - d_j(t)\}$ , где  $\pi$  есть некоторое расписание обслуживания требований множества  $N$  с момента времени  $t_0 = 0$ .

На рис. 7 представлены 3 из  $n!$  графиков функции  $F(\pi, t)$ . Определим функции  $\pi(t) = \operatorname{argmin}_{\pi} F(\pi, t)$  и  $F(t) = F(\pi(t), t) = \min_{\pi} F(\pi, t)$ .

Необходимо сделать следующие уточнения:

$F(\pi)$  (в качестве аргумента – расписание) обозначает суммарное запаздывание требований множества  $\{\pi\} \subseteq N$  при расписании  $\pi$ , упорядоченных с момента времени  $t_0 = 0$ ;

$F(t)$  (в качестве аргумента – момент времени, число  $t$ ) обозначает суммарное запаздывание требований множества  $\{\pi(t)\} \subseteq N$  с директивными сроками  $d_j(t)$ ,  $j = 1, 2, \dots, n$ , при расписании  $\pi(t)$ , упорядоченных с момента времени  $t_0 = 0$ .

Опишем некоторые свойства функции  $F(t)$ .

**Лемма 33.** Функция  $F(t)$  является кусочно-линейной, непрерывной, невозрастающей, неотрицательной.

**Доказательство.** Пусть  $\pi$  - некоторое расписание обслуживания требований множества  $N$  с момента времени  $t_0$ .

1. Покажем, что функция  $F(\pi, t)$  является непрерывной.

Пусть при  $t = t'$  для некоторого  $a$  для требований множества  $N' \subseteq N$ ,  $N' \neq \emptyset$ , выполняется  $L_j = c_j(\pi) -$

→  
t

Рис. 7. Функции  $F(\pi, t)$

$d_j(t') > a, \forall j \in N'$ , и пусть  $L' = \min_{j \in N'} L_j$ . Тогда при  $t \in (c_j(\pi) - d_j + d_n - a, c_j(\pi) - d_j + d_n + L)$  сохраняется соотношение  $c_j(\pi) - d_j(t) > 0, \forall j \in N'$ . Поэтому

$$\begin{aligned} F(\pi, t) &= \sum_{j \in N'} c_j(\pi) - d_j(t) = \\ &= \sum_{j \in N'} c_j(\pi) - (d_j - d_n + t), t \in [t', t' + L). \end{aligned}$$

Функция  $F(\pi, t)$  на данном интервале линейно зависит от  $t$  и, следовательно, непрерывна. Выбирая различные значения  $a$  мы получим кусочно-линейную непрерывную функцию на всей временной оси.

Таким образом, для каждой точки  $t \in R$  существует интервал вида  $[t, t + L)$  на котором функция  $F(\pi, t)$  непрерывна, а при расписании  $\pi$  запаздывает фиксированное количество требований.

Пусть числовая ось  $\mathbb{R}$  разбита на множество интервалов  $(-\infty, t_0), [t_0, t_1), \dots, [t_i, t_{i+1}), \dots, [t_m, +\infty)$ . На интервале  $[t_i, t_{i+1})$  при расписании  $\pi$  запаздывает  $k_i$  требований и функция  $F(\pi, t)$  непрерывна.

Покажем, что при увеличении  $t$  количество запаздывающих требований при расписании  $\pi$  не увеличивается. Для не запаздывающих требований выполняется  $c_j(\pi) - d_j + d_n - t \leq 0$ , то  $c_j(\pi) - d_j + d_n - (t + \Delta t) < 0, \forall \Delta t > 0$ .

Рассмотрим поведение функции в точке  $t_i, i = 0, 1, \dots, m$ . Пусть на интервале  $[t_{i-1}, t_i)$  при расписании  $\pi$  запаздывают требования множества  $N_1 \cup N_2$ , а на интервале  $[t_i, t_{i+1})$  запаздывают только требования множества  $N_2$ .

$$\begin{aligned}
F(\pi, t) &= \sum_{j \in N_1} (c_j(\pi) - (d_j - d_n + t)) + \\
&= \sum_{j \in N_2} (c_j(\pi) - (d_j - d_n + t)), t \in [t_{i-1}, t_i),
\end{aligned}$$

$$F(\pi, t) = \sum_{j \in N_1} (c_j(\pi) - (d_j - d_n + t)), t \in [t_i, t_{i+1}).$$

В точке  $t_i$  перестали запаздывать требования множества  $N_1$ , т.е.  $c_j(\pi) - (d_j - d_n + t_i) \leq 0, j \in N_1$ . Учитывая, что функция  $c_j(\pi) - (d_j - d_n + t)$  линейно зависит от  $t$ , тогда  $\lim_{t \rightarrow t_i-0} (c_j(\pi) - (d_j - d_n + t)) = 0, j \in N_1$ .

$$\begin{aligned}
&\lim_{t \rightarrow t_i-0} (\sum_{j \in N_1} (c_j(\pi) - (d_j - d_n + t)) + \sum_{j \in N_2} (c_j(\pi) - (d_j - d_n + t))) = \\
&\sum_{j \in N_1} \lim_{t \rightarrow t_i-0} (c_j(\pi) - (d_j - d_n + t)) + \lim_{t \rightarrow t_i-0} \sum_{j \in N_2} (c_j(\pi) - (d_j - d_n + t)) = \\
&0 + \lim_{t \rightarrow t_i-0} \sum_{j \in N_2} (c_j(\pi) - (d_j - d_n + t)) = \\
&\lim_{t \rightarrow t_i+0} \sum_{j \in N_2} (c_j(\pi) - (d_j - d_n + t)).
\end{aligned}$$

Следовательно функция  $F(\pi, t)$  непрерывна в каждой точке  $t_i$ .

2. Покажем, что функция  $F(\pi, t)$  является кусочно-линейной. Пусть на интервале  $[t, t + \Delta t]$  при расписании  $\pi$  запаздывает ровно  $k$  требований.

$$\begin{aligned}
F(\pi, t + \Delta t) - F(\pi, t) &= \sum_{j=1}^n \max\{0, c_j(\pi) - (d_j - d_n + t + \Delta t)\} - \\
&- \sum_{j=1}^n \max\{0, c_j(\pi) - (d_j - d_n + t)\}.
\end{aligned}$$

Учитывая, что только для  $k$  требований выполняется  $c_j(\pi) - (d_j - d_n + t + \Delta t) > 0$ :

$$F(\pi, t + \Delta t) - F(\pi, t) = F(\pi, t) - k\Delta t - F(\pi, t) = -k\Delta t.$$

Значит на любом интервале  $[t, t + \Delta t]$ , где фиксировано количество запаздывающих требований при расписании  $\pi$  функция задается уравнением  $F(\pi, t) = -kt + b$ , где  $k$  – количество запаздывающих требований.

3. Покажем, что функция  $F(\pi, t)$  является невозрастающей. Очевидно, что с ростом  $t$  значение  $\max\{0, c_j(\pi) - (d_j - d_n + t)\}$  не увеличивается. Таким образом функция  $F(\pi, t)$  является невозрастающей.
4.  $F(\pi, t) \geq 0, \forall t, \forall \pi$  из определения.

Следовательно, функция  $F(t) = \min_{\pi} F(\pi, t)$  непрерывная, кусочно-линейная, невозрастающая, неотрицательная.  $\square$

**Лемма 34.** Пусть числовая ось  $\mathbb{R}$  разбита на множество интервалов  $(-\infty, t_0), [t_0, t_1), \dots, [t_i, t_{i+1}), \dots, [t_m, +\infty)$ , где на интервале  $[t_i, t_{i+1})$  при расписании  $\pi(t)$  запаздывает  $k_i$  требований. Тогда на этом интервале  $F(t) = -k_i t + b_i, i = 1, \dots, m - 1$ . При этом  $b_0 = F(\pi_{SPT}, 0), b_{i+1} = b_i - k_i(t_{i+1} - t_i), b_m = 0$ .

**Доказательство.** Пусть на интервале  $[t, t + \Delta t]$  оптимальным является расписание  $\pi$  и при нем запаздывает ровно  $k$  требований. Тогда

$$F(t + \Delta t) - F(t) = \sum_{j=1}^n \max\{0, c_j(\pi) - (d_j - d_n + t + \Delta t)\} -$$

$$- \sum_{j=1}^n \max\{0, c_j(\pi) - (d_j - d_n + t)\}.$$

Учитывая, что только для  $k$  требований выполняется  $c_j(\pi) - (d_j - d_n + t + \Delta t) > 0$ :

$$F(t + \Delta t) - F(t) = F(t) - k\Delta t - F(t) = -k\Delta t.$$

Значит, на интервале  $[t, t + \Delta t]$  функция задается уравнением  $F(t) = -kt + b$ .

Покажем, что  $b_0 = F(\pi_{SPT}, 0)$ . Так как в точке  $t = 0$  запаздывают все требования при любом расписании  $\pi$ , то оптимальным расписанием в этой точке будет  $\pi_{SPT}$ .

В точке  $t_m$  при оптимальном расписании перестают запаздывать все требования, т.е.  $F(t) = 0, \forall t \geq t_m$ , следовательно  $b_m = 0$ .

Учитывая, что функция  $F(t)$  кусочно-линейная и непрерывная:  $b_{i+1} = b_i - k_i(t_{i+1} - t_i), i = 0, 1, \dots, m - 1. \square$

Для графика кусочно-линейной функции  $F(t)$  выделены 2 типа точек "излома": точки типа  $a$  и  $b$  (рис. 8).

В точках типа  $b$  происходит изменение оптимального расписания.

**Лемма 35.** Точки типа  $a$  соответствуют ситуации, когда сократилось количество запаздывающих требований.

**Доказательство.** Рассмотрим интервал  $[t, t + \Delta t]$ . Пусть точка  $t'$ , точка типа  $a$ , принадлежит интервалу, и других точек "излома" графика функции  $F(t)$  на интервале  $[t, t + \Delta t]$  нет. Пусть на интервале  $[t, t')$  при расписании  $\pi(t)$  запаздывает  $k$  требований.

Из рис. 8 видно, что имеет место следующее неравенство:

$$F(t + \Delta t) > F(t) - k\Delta t.$$

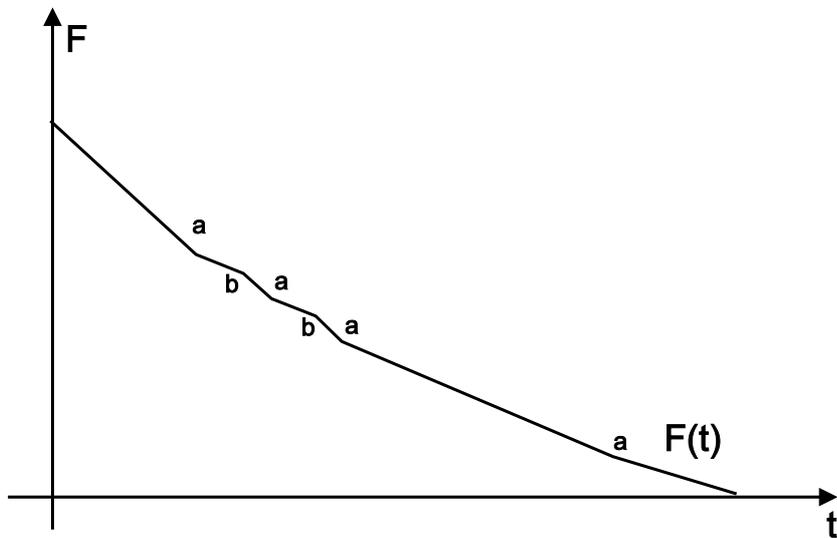


Рис. 8. Точки типа  $a$  и  $b$

Пусть на интервале  $[t', t + \Delta t]$  при данном расписании запаздывает  $m$  требований. Покажем, что  $m < k$ .

От противного, если  $m \geq k$ , то  $F(t + \Delta t) = F(t') - m(t + \Delta t - t') = F(t) - k(t' - t) - m(t + \Delta t - t') \leq F(t) - k(t' - t) - k(t + \Delta t - t') = F(t) - k\Delta t$ .

Получили противоречие, следовательно  $m < k$ .  $\square$

### 6.1. Свойства функции $F_l(t)$

Функция  $F_l(t)$ , построенная на шаге  $l$  алгоритма **В-1**, является функцией  $F(t)$  определенной ранее, так как функция  $F_l(t)$  соответствует оптимальному частичному расписанию  $\pi_l(t)$ , построенному из требований множества  $N_l = \{l, l+1, \dots, n\}$ . Для упрощения записи будем обозначать функцию  $F_l(t)$  как  $F(t)$ . Функция определена на  $R$  следующим образом:

$$F_l(t) = -(n - (l - 1))t + F_l(0), t \leq 0;$$

$$F_l(t) \text{ определяется согласно алгоритму В-1,} \\ t \in [0, \sum_{i=1}^n p_i];$$

$$F_l(t) = 0, t \geq \sum_{i=1}^n p_i.$$

**Лемма 36.** Функция  $F_l(t)$  является кусочно-линейной.

**Доказательство.** Доказательство проведем методом математической индукции. На шаге  $l = n$ , очевидно, построена кусочно-линейная функция  $F_n(t) = \max\{0, p_n - t\}$ . Предположим, что на шаге  $l+1$  построена кусочно-линейная функция.

Рассмотрим шаг  $l$ . При  $t \notin [0, \sum_{j=l}^n p_j]$  функция  $F_l(t)$  линейно зависит от аргумента  $t$  (см. **алгоритм В-1**).

Рассмотрим интервал  $t \in [0, \sum_{j=l}^n p_j]$ . Функция  $F(\pi^1, t)$  построена из функции  $F_{l+1}(t)$  линейными преобразованиями (сдвигом "вправо" на величину  $p_l$  и "прибавлением" кусочно-линейной функции  $\max\{0, p_l - d_l(t)\}$ , которая линейно зависит от  $t$ ). Так как  $F_{l+1}(t)$  кусочно-линейная, то функция  $F(\pi^1, t)$  является кусочно-линейной. Аналогично функция  $F(\pi^2, t)$  является кусочно-линейной. Следовательно  $F_l(t) := \min\{F(\pi^1, t), F(\pi^2, t)\}$  кусочно-линейная.  $\square$

**Лемма 37.** Для случая (2) в точках типа **a** происходит сокращение количества запаздывающих требований на единицу (или на два, если продолжительность второго по порядку запаздывающего требования равно  $p_n$ , его директивный срок равен  $d_n$ , а  $d_n - d_1 = p_n$ ).

**Доказательство.** Рассмотрим расписание  $\pi = (\dots, j_1, j_2, \dots, j_k)$  в точке  $t$ , где  $j_1$  - первое запаздывающее требование. Пусть  $c_{j_1}(\pi) - d_{j_1}(t) = \Delta t$ .

$$\begin{aligned} c_{j_2}(\pi) - d_{j_2}(t) &= c_{j_1}(\pi) + p_{j_1} - d_{j_2}(t) \geq c_{j_1}(\pi) + p_n - d_{j_2}(t) = \\ &= d_{j_1}(t) + \Delta t + p_n - d_{j_2}(t). \end{aligned}$$

Учитывая, что  $p_n \geq d_k(t) - d_l(t)$ ,  $\forall k, l$ , то  $c_{j_2}(\pi) - d_{j_2}(t) = \Delta t$  только, если  $p_{j_2} = p_n$ ,  $d_{j_2} - d_{j_1} = p_n \Rightarrow d_n - d_1 = p_n$  и  $d_{j_2} = d_n$ , иначе  $c_{j_2}(\pi) - d_{j_2}(t) > \Delta t$ .

Но  $c_{j_3}(\pi) - d_{j_3}(t) = c_{j_1}(\pi) + p_{j_2} + p_{j_3} = d_{j_1}(t) + \Delta t + p_{j_2} + p_{j_3} > \Delta t$  (аналогично для остальных запаздывающих требований  $j_4, \dots, j_k$ ).

Тогда при  $t \rightarrow t + \Delta t$  в расписании  $\pi$  перестанет запаздывать только 1 требование (или же 2 требования, если  $p_{j_2} = p_n$ ,  $d_{j_2} = d_n$ ,  $d_n - d_1 = p_n$ ).  $\square$

Можно предположить, что в точке типа **b** изменение количества запаздывающих требований также происходит на единицу для случая (2).

Рис. 9. Запаздывание требований

**Лемма 38.** Для случая (2) существует такое оптимальное расписание  $\pi^*$ , что  $c_n(\pi^*) \in (d_n - p_l - p_n, d_n + p_l + p_n)$ , где  $l$  – первое запаздывающее требование (т.е. требование  $n$  "находится" в оптимальном расписании "около"  $d_n$ ). И, если требование  $n$  запаздывает, то является либо первым, либо вторым запаздывающим требованием.

**Доказательство.**

1. Предположим, что построенное расписание имеет вид:  $\pi_1 = (\pi^1, n, \pi^2, l, \pi^3)$ , где  $l$  – первое запаздывающее требование.

а) Если  $c_l(\pi_1) - d_l > p_n$ , то  $c_l(\pi_1) - d_n \leq p_n$ . Рассмотрим расписание  $\pi_2 = (\pi^1, \pi^2, l, n, \pi^3)$ . В расписании  $\pi_2$  требование  $l$  также запаздывает, так как

$$c_l(\pi_2) = c_l(\pi_1) - p_n > d_l.$$

Выполняется

$$c_n(\pi_2) = c_l(\pi_1),$$

$$\begin{aligned} \text{тогда } F(\pi_1) - F(\pi_2) &= (c_l(\pi_1) - d_l) - (c_l(\pi_2) - d_l + c_n(\pi_2) - d_n) \\ &= (c_l(\pi_1) - d_l) - (c_l(\pi_2) - d_l + c_l(\pi_1) - d_n) = -(c_l(\pi_2) - d_n) \\ &= -(c_l(\pi_1) - p_n - d_n) \geq 0, \end{aligned}$$

т.е. расписание  $\pi_2$  не увеличивает значение суммарного запаздывания. В этом случае  $c_n(\pi_2) \leq d_n + p_n$ ;

б) Если  $c_l(\pi_1) - d_l > p_n$ , то  $c_l(\pi_1) - d_n > p_n$ . При расписании  $\pi_2 = (\pi^1, \pi^2, n, l, \pi^3)$  значение целевой функции не изменится, так как  $n$  - по-прежнему не запаздывающее требование (см. доказательство леммы 10). В этом случае  $c_n(\pi_2) > d_n - p_n - p_l$ ;

в)  $c_l(\pi_1) - d_l \leq p_n$ . Рассмотрим расписание  $\pi_2 = (\pi^1, \pi^2, l, n, \pi^3)$ . В расписании  $\pi_2$  требование  $l$  не запаздывает, так как  $c_l(\pi_2) = c_l(\pi_1) - p_n \leq d_l + p_n - p_n = d_l$ .  $c_n(\pi_2) = c_l(\pi_1)$ . Если требование  $n$  запаздывает при расписании  $\pi_2$ :

$$F(\pi_1) - F(\pi_2) = (c_l(\pi_1) - d_l) - (c_n(\pi_2) - d_n) = d_n - d_l \geq 0,$$

иначе, если требование  $n$  не запаздывает, то  $F(\pi_1) - F(\pi_2) = (c_l(\pi_1) - d_l) \geq 0$ , т.е. расписание  $\pi_2$  не увеличивает значение суммарного запаздывания. В этом случае  $c_n(\pi_2) \leq d_n + p_n$ ;

2. Предположим, что построенное расписание имеет вид  $\pi_1 = (\pi^1, l, \pi^2, n, \pi^3)$ , где  $l$  - первое запаздывающее требование. Очевидно, что после требования  $l$  все требования запаздывают. В оптимальном расписании запаздывающие требования после  $l$  будут упорядочены в порядке SPT. Требование  $n$  окажется вторым запаздывающим после  $l$ , т.е. расписание  $\pi_2 = (\pi^1, l, n, \pi^2, \pi^3)$  будет оптимальным. В этом случае  $c_n(\pi_2) < d_n + p_n + p_l$ .  $\square$

Определим интервалы  $[t_k, T_k)$  следующим образом:

$$t_k = \sum_{i=k+1}^n p_i, \quad T_k = \sum_{i=1}^{n-k+1} p_i + d_n - d_1$$

**Лемма 39.** В точке  $t \notin [t_k, T_k)$  при любом расписании  $\pi$  количество запаздывающих требований не равно  $k$ .

**Доказательство.** Покажем, что если  $t > T_k$ , то при любом расписании запаздывает меньше  $k$  требований.

Рассмотрим произвольное расписание  $\pi = (\dots, j_0, j_1, \dots, j_k)$ . Пусть  $j_1$  – первое запаздывающее требование. Причем требование  $j_0$  не запаздывает. В лемме 9 было доказано, что если при расписании запаздывает требование  $j_1$ , то последующие требования так же запаздывают. Аналогично, если требование  $j_1$  не запаздывает, то не запаздывают и предшествующие требования.

Тогда незапаздывающих требований будет  $n - k$ . Обозначим множество незапаздывающих требований в расписании  $\pi$  через  $N'$ ,  $|N'| = n - k$ . Рассмотрим запаздывание требования  $j_1$ .

$$T_{j_1}(\pi) = \max\{0, \sum_{j \in N'} p_j + p_{j_1} - d_{j_1} + d_n - t\} > 0.$$

Если  $t > T_k$ , тогда выполняется

$$t > T_k = \sum_{i=1}^{n-k+1} p_i + d_n - d_1 \geq \sum_{j \in N'} p_j + p_{j_1} + d_n - d_{j_1}.$$

При этом требование  $j_1$  не запаздывает. Следовательно, если  $t > T_k$ , то при любом расписании запаздывает меньше  $k$  требований.

$$T_{j_0}(\pi) = \max\{0, P(N') - d_{j_0} + d_n - t\}.$$

Если  $t < t_k = \sum_{i=k+1}^n p_i$ , тогда выполняется

$$P(N') - d_{j_0} + d_n - t > P(N') - d_{j_0} + d_n - \sum_{i=k+1}^n p_i \geq 0.$$

Следовательно, требование  $j_0$  запаздывает. Поэтому при любом расписании  $\pi$  запаздывает больше  $k$  требований.  $\square$

Стоит выделить несколько свойств этих интервалов.

1.  $T_{k+1} < T_k$ ,  $k = 0, 1, \dots, n-1$ , так как  $T_k - T_{k+1} = (\sum_{i=1}^{n-k+1} p_i + d_n - d_1) - (\sum_{i=1}^{n-k} p_i + d_n - d_1) = p_{n-k+1} > 0$ ;
2.  $t_{k+1} < t_k$ , т. к.  $t_k - t_{k+1} = \sum_{i=k+1}^n p_i - \sum_{i=k+2}^n p_i = p_{k+1} > 0$ ;
3.  $t_k < T_k$ ;
4. Очевидно, что существуют интервалы  $[\bar{t}_1, \bar{t}_2]$  на которых может запаздывать не только  $k$  требований, но не существует интервалов, в которых количество запаздывающих требований не определено.
5. **Лемма 40.** Если  $T_{k+1} < t_{k-1}$ , то на промежутке  $(T_{k+1}, t_{k-1})$  при любом расписании может запаздывать только  $k$  требований (рис.10).

**Доказательство.** Предположим, что для  $t \in (\bar{t}_1, \bar{t}_2) \subset (T_{k+1}, t_{k-1})$  может запаздывать  $l \neq k$  требований. Тогда должно выполняться:  $T_{k+1} < T_l$  и  $t_l < t_{k-1}$ , что противоречит сочетанию свойств **1** и **2**.  $\square$

Опишем ситуации, когда может выполняться  $T_{k+1} < t_{k-1}$ .

а) Если  $k \leq n/2$ , тогда:

$$\sum_{i=1}^{k-1} p_i + d_n - d_1 < \sum_{i=n-k+1}^n p_i$$

при этом  $d_n - d_1 \geq 0$ , т.е. продолжительность требования  $(n-k+1) p_{n-k+1}$  "покрывает" разницу  $(k-1)$

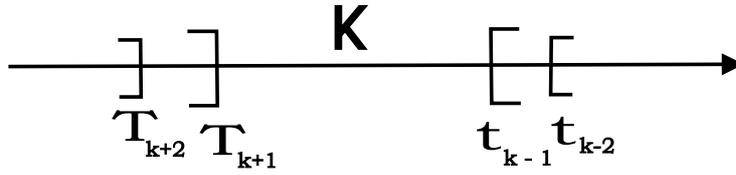


Рис. 10. Известное количество запаздывающих требований

самых *больших* (продолжительных)  $\sum_{i=1}^{k-1} p_i$  и  $(k-1)$  самых *маленьких*  $\sum_{i=n-k}^n p_i$  требований.

б) Если  $k > n/2$ , тогда

$$\sum_{i=1}^{n-k} p_i + d_n - d_1 < \sum_{i=k}^n p_i.$$

Также одно требование с продолжительностью  $p_k$  "покрывает" разницу  $(n-k)$  самых больших  $\{1, 2, \dots, n-k\}$  и  $(n-k)$  самых маленьких по продолжительности требований  $\{k+1, \dots, n\}$ .

Интервалы, в которых запаздывает точно  $k$  требований, могут возникнуть, если  $p_1 - p_n < p_{n-1}$ .

6. Расстояние между точками  $T_k$  и  $T_{k+1}$ :  $T_k - T_{k+1} = \sum_{i=1}^{n-k+1} p_i + d_n - d_1 - (\sum_{i=1}^{n-k} p_i + d_n - d_1) = p_{n-k+1}$ ;
7. Учитывая, что для любого  $k$  существует  $l$ , для которого выполняется  $t_{l+1} \leq T_k \leq t_l$ , поэтому имеет место  $T_k - t_l \leq p_{l+1}$ , так как  $t_l - t_{l+1} = p_{l+1}$ ;
8. Если  $t_1 < T_n$ , т.е.  $\sum_{i=2}^n p_i < p_1 + d_n - d_1$ , то на интервале  $[t_1, T_n)$  могут запаздывать от 1 до  $n$  требований;

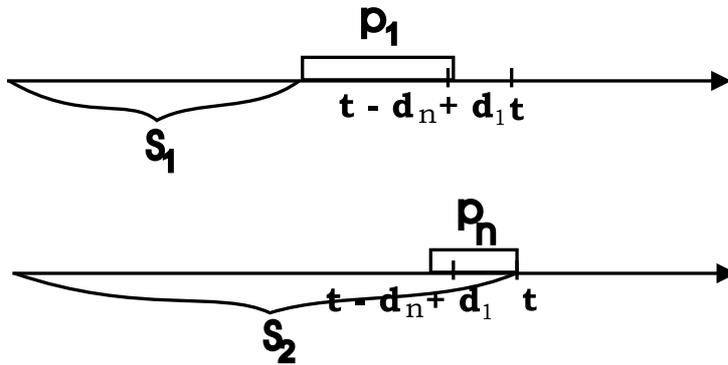


Рис. 11. Положение незапоздывающих требований.

9. **Лемма 41.** Пусть в точке  $t$  может запаздывать  $k$  требований. То существует такое расписание  $\pi$ , при котором в данной точке запаздывает ровно  $k$  требований.

**Доказательство.** Поставим вопрос: существует ли расписание  $\pi$ , при котором в точке  $t$  не запаздывает ровно  $n - k$  требований? Множество незапоздывающих требований обозначим  $J = (j_1, j_2, \dots, j_{n-k})$ .

Чтобы в расписании не запаздывало  $n - k$  требований необходимо, чтобы  $S(J) \in (S_1, S_2] = (t - (d_n - d_1) - p_1, t]$  (см. рис.11), где  $S(J) = \sum_{i \in J} p_i$ . Вопрос, можно ли так выбрать  $n - k$  требований?

Очевидно, что при  $(S_1, S_2] \subset [0, \sum_{i=k+1}^n p_i) \cup (\sum_{i=1}^{n-k} p_i, \sum_{i=1}^n p_i]$  ответ "НЕТ".

Рассмотрим случай, когда  $(S_1, S_2] \subset [\sum_{i=k+1}^n p_i, \sum_{i=1}^{n-k} p_i]$ .

Организуем следующий перебор.

**Шаг 0.**  $J = (k+1, k+2, \dots, n)$ , таким образом найдена наименьшая сумма  $S(J)$ .

**Основной шаг.** Выбираются такие требования  $j_1 \in J$  и  $i \notin J$ , что  $p_{j_1} < p_{j_2}$ . Требования выбираются исходя из  $\min_{i \in J'} \{p_i - p_{j_1}\}$ .  $J' := i \cup J \setminus j_1$ . Очевидно, что  $S(J') > S(J)$ . Для этого необходимо сделать  $(n-k)k$  сравнений. Так мы обеспечим последовательный перебор  $S(J^1) < S(J^2) < \dots < S(J^m)$  не пропустив ни одно значение  $S$ .

В какой-то момент окажется  $S(J) \in (S_1, S_2]$ .

Докажем это от противного. Пусть  $S(J^l) \leq S_1$  на некотором шаге  $l$  и на шаге  $l+1$ :  $S(J^{l+1}) > S_2$ , т.е. оказалось, что  $\min\{p_{j_2} - p_{j_1}\} > S_2 - S_1 = p_1 + (d_n - d_1)$ , что невозможно.

Таким образом, существует расписание  $\pi$ , при котором в точке  $t$  запаздывает  $k$  требований  $\square$

## 6.2. Интерполяционные оценки функции $F(t)$

Для случая (2) предлагается интерполяционный подход к решению. На каждом шаге  $l$  алгоритма **В-1** вместо точной функции  $F_l(t)$  строится приближенная функция  $\Phi(t)$ . В точках итерполирования  $\Phi(t) = F_l(t)$ . Предлагается использовать значения  $\Phi(t)$  вместо значений  $F_l(t)$  на шаге  $l - 1$  алгоритма **В-1**.

Необходимо выбрать точки и функцию интерполирования  $\Phi(t)$ , оценить погрешность функции  $\Phi(t)$ .

В данной работе в качестве точек интерполирования выбрано множество точек  $t_k$  и  $T_k$  (учитывая  $t_0 = T_1$ , здесь  $t_0$  – не время начала обслуживания требований). Также берется точка  $t = 0$ . Количество точек не превышает  $2n$ . Расстояние между точками интерполирования, как следует из свойств **6** и **7**, не превышает продолжительности одного требования, т.е. интервал  $[0, \sum_{j=1}^n p_j]$  покрывается "равномерно".

Рассмотрим произвольный интервал  $[\bar{t}_1, \bar{t}_2]$ , где  $\bar{t}_1, \bar{t}_2 \in \{t_1, t_2, \dots, t_n, T_1, T_2, \dots, T_n\}$ . Пусть в точках  $\bar{t}_1$  и  $\bar{t}_2$  мы знаем точное значение  $F_l(t)$  и соответствующие оптимальные расписания  $\pi_1$  и  $\pi_2$ . Восстановим графики  $F(\pi_1, t)$  и  $F(\pi_2, t)$  для  $t \in [\bar{t}_1, \bar{t}_2]$ , см. рис. 12. Это можно сделать за полиномиальное число шагов, восстановив не более  $n$  точек типа  $a$  для каждого графика и соответствующую точку  $b$  – их пересечение.

**Лемма 42.** График функции  $\min\{F(\pi_1, t), F(\pi_2, t)\}$  является верхней границей для графика точной функции  $F_l(t)$  на интервале  $[\bar{t}_1, \bar{t}_2]$ .

**Доказательство.** Предположим обратное. Пусть в точке  $t \in [\bar{t}_1, \bar{t}_2]$  оптимальным является расписание  $\pi^*$ . Причем  $F(t, \pi^*) > \min\{F(\pi_1, t), F(\pi_2, t)\}$ , т.е. расписание соответствующее  $\min\{F(\pi_1, t), F(\pi_2, t)\}$  лучше чем  $\pi^*$ .

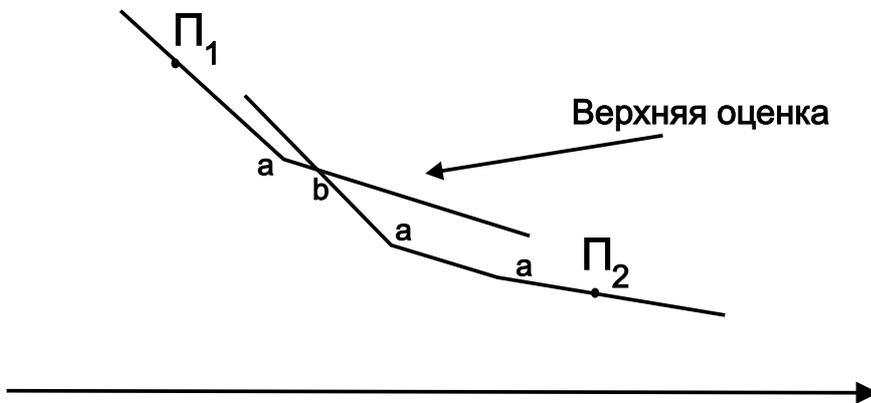


Рис. 12. Интерполяционная функция

Получили противоречие. Значит график функции  $\min\{F(\pi_1, t), F(\pi_2, t)\}$  на  $[\bar{t}_1, \bar{t}_2]$  является верхней границей для графика точной функции  $F_l(t)$ .  $\square$

В качестве интерполяционной функции  $\Phi(t)$  выберем верхнюю границу.

### 6.3. Погрешность приближенной функции $\Phi(t)$

**Лемма 43.** Если на некотором интервале  $[\bar{t}_1, \bar{t}_2]$  может запаздывать  $k$  и  $k + m$  требований, то может запаздывать и  $k + 1, k + 2, \dots, k + m - 1$  требований.

**Доказательство.** Выполняется  $t_{k+m} < t_k < \bar{t}_1 < \bar{t}_2 < T_{k+m} < T_k$ . Учитывая свойства **1** и **2**:  $t_{k+m} < t_{k+m-1} < \dots < t_{k+1} < t_k < \bar{t}_1 < \bar{t}_2 < T_{k+m} < T_{k+m-1} < \dots < T_{k+1} < T_k$ , то на промежутке  $[\bar{t}_1, \bar{t}_2]$  может запаздывать и  $k + 1, k + 2, \dots, k + m - 1$  требований.  $\square$

**Лемма 44.** Пусть известно, что на промежутке  $[\bar{t}_1, \bar{t}_2]$  может запаздывать  $\{k, k + 1, \dots, k + m\}$  требований. "Грубая" нижняя граница строится комбинацией наклонов графика функции  $k_{max} = \max\{k, k + 1, \dots, k + m\} = k + m$  и  $k_{min} = \min\{k, k + 1, \dots, k + m\} = k$  (рис. 13)

**Доказательство.** Предположим, что график  $F_1(t) = -k_1t + b_1$  проходит ниже графика функции  $F_{max}(t) = -K_{max}t + b$  (см рис. 14). Это возможно только, если  $-k_1 < -K_{max}$ , т.е.  $K_{max} < k_1$ , что невозможно. Аналогично можно показать, что график функции  $F(t)$  проходит выше (на заданном интервале), чем график функции  $F_{min}$ .  $\square$

Можно построить более точную нижнюю границу. Так как в каждой точке "излома" типа **a** происходит изменение количества запаздывающих требований лишь на 1 (лемма 37), то резкий переход от  $k_{max}$  к  $k_{min}$  невозможен, если  $k_{max} - k_{min} > 1$ .

Более точная нижняя граница (рис. 15) имеет  $k_{max} - k_{min}$  изломов типа **a**, причем расстояние между ними превышает  $p_n$  (чтобы в расписании последовательно стало не запаздывать два требования, необходимо чтобы  $t$  изменилось более чем на длину одного требования). Следовательно, если длина интервала  $[\bar{t}_1, \bar{t}_2]$  меньше  $p_n$  то там возможен лишь 1 переход типа **a** (имеется в виду, что все точки **a** из

Рис. 13. Грубая нижняя граница

Рис. 14. Док-во "грубой" нижней границы

Рис. 15. Более точная нижняя граница

*этого промежутка характеризуют переход от наклона  $k$  к наклону  $k - 1$ ).*

Оценить максимальное расстояние между нижней и верхней границей легче, если использовать "грубую" нижнюю границу.

Можно оценить расстояние  $h$  (рис.16), характеризующую абсолютную погрешность целевой функции. Пусть мы знаем координаты точек  $(\bar{t}_1, F_1)$  и  $(\bar{t}_2, F_2)$  (далее  $(x_1, y_1)$  и  $(x_2, y_2)$ ).

Отрезок, соединяющий эти точки, так же можно воспринимать как некую верхнюю оценку. Несложно вычислить наклон  $k$  этого отрезка:

$$k = \frac{y_2 - y_1}{x_2 - x_1}.$$

$$h = dt(k_{max} - k) = (x_2 - x_1 - dt)(k - k_{min}).$$

Рис. 16. Расстояние между границами

$$dt = \frac{(x_2 - x_1)(k - k_{min})}{(k_{max} - k_{min})}, \quad h = \frac{(x_2 - x_1)(k_{max} - k)(k - k_{min})}{(k_{max} - k_{min})}.$$

Если мы точно знаем количество запаздывающих на промежутке  $(\bar{t}_1, \bar{t}_2)$  требований  $k_{max} = k_{min} = k$ , то верхняя граница является точным решением.

Если на промежутке может запаздывать только  $k$  или  $k - 1$  требований, то расстояние между верхней и нижней границей можно вычислить иначе (рис. 17).

Рис. 17. Если может запаздывать только  $k$  или  $k - 1$  требований

#### 6.4. Аппроксимационная схема решения задач теории расписаний

В данном подразделе представлен подход к решению задач теории расписаний, основанный на аппроксимации целевой функции при изменении параметров требований. Данный подход опишем на основе исследуемой функции минимизации суммарного запаздывания.

Рассмотрим произвольный пример проблемы  $\langle \{p_j, d_j\}_{j \in N}, 0 \rangle$  с параметрами требований  $p_j, d_j, j \in N$ . Пусть  $\pi^*$  – оптимальное расписание обслуживания требований для данного примера. Если мы будем изменять параметры требований, то очевидно, значение целевой функции также будет меняться. Рассмотрим случай, когда изменение параметров требований осуществляется с помощью одной переменной  $\lambda$ . Например,  $p_j(\lambda) = \lambda p_j$  или  $d_j(\lambda) = \lambda d_j, \forall j$ . Тогда целевая функция будет являться функцией от одного параметра  $\lambda$ , а  $\pi^*(\lambda)$  – оптимальные расписания для соответствующего  $\lambda$ , т.е.  $F(\lambda) = F(\pi^*(\lambda))$ .

Если мы не можем решить исходный пример, т.е. при  $\lambda = 1$ , за "разумное" время (или за "разумный" объем операций компьютера), но мы можем решить примеры с измененными параметрами требований, то мы можем попытаться найти приближенное значение целевой функции исходного примера с помощью интерполирования. Пусть модифицированный пример  $\langle \{\lambda p_j, d_j\}_{j \in N}, 0 \rangle$ ,  $\lambda \in (0, +\infty)$ . Решением исходного примера будет  $F(1)$ .

Функция  $F(\lambda)$  в данном случае является вогнутой, возрастающей, монотонной и кусочно-линейной с не более чем  $n$  точками "перегиба".

В алгоритме **В-1** мы изменяли директивные сроки  $d_j(t) = d_j - d_n + t$ . Причем на краях (концах) исследуемого интервала  $[0, \sum p_j]$  задачи решаются не более чем за  $O(n \log n)$  операций. Для достаточно малого  $\lambda_a$  ( $\lambda_a < \frac{d_{min}}{\sum p_j}$ ) оптимальным будет любое из  $n!$  расписаний. Для достаточно большого  $\lambda_b$  ( $\lambda_b < \frac{d_{max}}{p_{min}}$ ) оптимальным будет *EDD*-расписание. Таким образом, мы можем интерполировать целевую функцию на интервале  $[\lambda_a, \lambda_b]$ . Точками интерполирования можно взять корни полинома Чебышева. Одной из особенностей корней полинома Чебышева является то, что они расположены на концах интервала, где трудоемкость решения примеров существенно меньше,  $\lambda_1, \dots, \lambda_m \in [\lambda_a, \lambda_b]$ . На рис. 18 показан график изменения сложности примеров (количество вершин в дереве поиска алгоритма А) и изменение значения целевой функции относительно изменения параметра  $\lambda$ . Интерполяционный полином Лагранжа строится по точкам  $(\lambda_1, F(\lambda_1)), \dots, (\lambda_m, F(\lambda_m))$ . Во всех точках интерполирования  $H(\lambda_i) \leq \bar{H}$ ,  $i = 1, \dots, m$ , т.е. сложность примеров не превышает "порогового" значения трудоемкости  $\bar{H}$ .

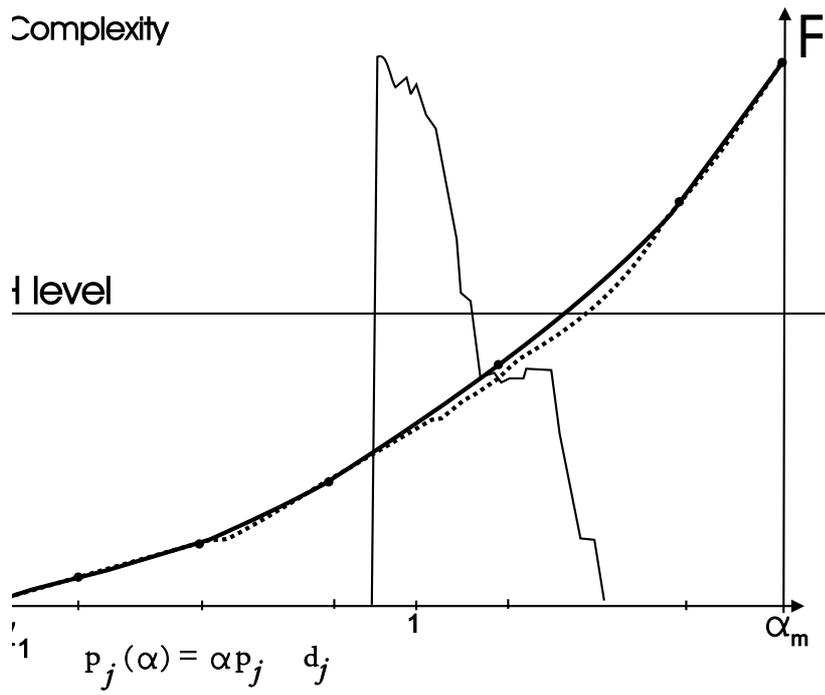


Рис. 18. Аппроксимационная схема решения задачи

## 7. Метаэвристический подход

Нами построен **гибридный алгоритм**, использующий идею известного метаэвристического **алгоритма "Муравьиные колонии" (АСО)** [14] и комбинаторные свойства правил исключения 1-4. В разд. 7.1. приводится **алгоритм АСО**. Далее описывается **гибридный алгоритм** и приводится сравнительный анализ его эффективности и эффективности **алгоритма АСО**.

### 7.1. Алгоритм АСО для задачи $1||\sum T_j$

Для решения многих комбинаторных задач эффективно используются *метаэвристические* методы: **генетические алгоритмы**, **локальный поиск с запретами (Tabu search)**, **муравьиные колонии (Ant Colony Optimization)** и др.

В данной работе рассматривается метод Ant Colony Optimization (АСО) [14]. Этот итерационный метод основан на идее последовательного приближения к оптимальному решению.

Каждая итерация – "запуск искусственного муравья", который "пытается" по некоторому правилу выбрать наилучший маршрут к "пище" (к оптимуму функции) используя метки своих предшественников.

Каждый муравей выполняет цепочку шагов. На каждом шаге  $i = 1, 2, \dots, n$  выбирается требование  $j$  для подстановки на место  $i$  расписания используя "вероятности перехода".

В алгоритме используются параметры:

$\eta_{ij}$  – эвристическая информация о том, насколько хорошим кажется постановка требования  $j$  на место  $i$  в

расписании. Этот параметр вычисляется эвристически по одному из правил.

1. EDD:  $\eta_{ij} = \frac{1}{d_j}$ ;
2. MDD (modified due date). В алгоритме MDD последовательно на позиции  $i = 1, \dots, n$  выбирается еще неупорядоченное требование  $j$  с наименьшим значением  $\max\{S + p_j, d_j\}$ , где  $S$  – сумма продолжительностей предшествующих упорядоченных требований. Эвристическая информация рассчитывается следующим образом:  

$$\eta_{ij} = \frac{1}{\max\{S+p_j, d_j\}}$$
;
3. L-MDD (Look-ahead MDD):  $\eta_{ij} = \frac{1}{Tard_j}$ ,  
 где  $Tard_j$  – суммарное запаздывание при модифицированном расписании MDD, при котором на позиции  $i$  обслуживается требование  $j$ , а все остальные требования упорядочены в соответствии с правилом MDD;
4. SPT:  $\eta_{ij} = \frac{1}{p_j}$ ,

$\tau_{ij}$  – "след" (в природе: след феромона). После каждой итерации этот параметр корректируется. Параметр показывает насколько "хорошим" для требования  $j$  оказалась позиция  $i$ . То есть это накопленная статистическая информация о качестве выбора для позиции  $i$  требования  $j$ , в то время как  $\eta_{ij}$  характеризует предполагаемую выгоду такой подстановки при недостатке накопленной информации.

Перед первой итерацией  $\tau_{ij} = 1/(mT_{EDD})$ , где  $T_{EDD}$  – суммарное запаздывание при EDD-расписании.  $m$  – количество итераций (муравьев). Параметры  $\eta_{ij}$  рассчитываются один раз перед первой итерацией.

На каждом шаге вычисляется **матрица вероятностей перехода**:

$$\rho_{ij} = \begin{cases} \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{h \in \Omega} [\tau_{ih}]^\alpha [\eta_{ih}]^\beta}, & j \in \Omega, \\ 0, & j \notin \Omega, \end{cases}$$

где  $\Omega$  – множество неупорядоченных требований.

Правило, по которому на позицию  $i$  выбирается требование  $j$ , определяется следующим образом:

$$\begin{cases} j = \mathit{argmax}_{h \in \Omega} [\tau_{ih}]^\alpha [\eta_{ih}]^\beta, & q < q_0, \\ j \text{ определяется случайным образом} \\ \text{согласно распределению вероятностей } \rho_{ij}, & q \geq q_0, \end{cases}$$

где  $0 \leq q_0 \leq 1$  – параметр алгоритма, а значение  $q$  вычисляется случайным образом на каждом шаге.

После того, как требование  $j$  было поставлено на позицию  $i$ , пересчитывается "локальный след":

$$\tau_{ij} = (1 - \rho)\tau_{ij} + \rho\tau_0,$$

где  $\tau_0 = 1/(mT_{EDD})$ .  $T_{EDD}$  – суммарное запаздывание при EDD-расписании.  $\rho \in [0, 1]$  – коэффициент распада феромона (параметр алгоритма).

После каждой итерации "глобальный след"  $\tau_{ij}$  корректируется по правилу

$$\tau_{ij} = (1 - \rho)\tau_{ij} + \rho/T^*,$$

если в "лучшем" найденном расписании на позиции  $i$  обслуживается требование  $j$ . Иначе

$$\tau_{ij} = (1 - \rho)\tau_{ij},$$

Значение  $T^*$  – суммарное запаздывание для "лучшего" найденного расписания.

**Алгоритм АСО** дает хорошие результаты при интеграции в него локального поиска, к примеру, попарной перестановки требований. Локальный поиск запускается после каждой итерации.

В работе [14] приводится экспериментальная оценка эффективности этого алгоритма. Эксперименты проводились для тестовых примеров [9] при  $n = 50, 100$ . Для размерности задачи  $n = 50$  из 125 примеров неточно был решен только 1. Для размерности  $n = 100$  неточно решено 0 примеров из 125. Относительная погрешность не превосходила 0.08%.

В своих экспериментах мы использовали те же значения управляющих параметров и эвристики, как и в работе [14].

Нетрудно убедиться, что трудоемкость алгоритма без локального поиска  $O(mn^2)$ . Для каждой позиции  $i$  (всего  $n$  позиций) выбиралось требование  $j$  за  $O(n)$  операций.

Трудоемкость локального поиска  $O(n^3)$ . Тогда трудоемкость **алгоритма АСО** с локальным поиском составляет не меньше  $O(mn^3)$  операций.

## 7.2. Гибридный алгоритм

На основе алгоритма АСО и правил исключения 1-4 нами построен новый гибридный алгоритм.

На каждой итерации запускается модифицированный алгоритм **A**, в котором очередное требование  $j^*$  "случайным" образом ставится на некоторую позицию  $k \in L(N, t)$ .

**Модифицированная процедура ProcL** ( $N, t$ )

0. Дан пример  $\{N, t\}$  с множеством требований  $N = \{j_1, j_2, \dots, j_n\}$  и моментом начала обслуживания  $t$ ,  $d_{j_1} \leq d_{j_2} \leq \dots \leq d_{j_n}$ ;
1. **IF**  $N = \emptyset$  **THEN**  $\pi^* :=$  пустое расписание, **GOTO 7.**;
2. Найдем требование  $j^*(N, t)$  из множества  $N$ ;
3. Найдем множество  $L(N, t)$  для требования  $j^*$ ;
4. Рассчитаем матрицу вероятностей перехода для каждой позиции  $i \in L(N, t)$ :

$$\rho_{ij^*} = \frac{\tau_{ij^*} 1/F(\pi^i)}{\sum_{h \in L(N, t)} \tau_{hj^*} 1/F(\pi^h)},$$

где  $\pi^i = (j_1, \dots, j_{m-1}, j_{m+1}, \dots, j_i, j^*, j_{i+1}, \dots, j_n)$ ,  $j^* = j_m$ ,  $m < i$ ;

5. Выберем позицию  $k \in L(N, t)$  произвольным образом согласно распределению вероятностей  $\rho_{ij^*}$ ;
6. Пересчитаем "локальный след":

$$\tau_{kj^*} = (1 - \rho)\tau_{kj^*} + \rho\tau_0,$$

где  $\tau_0 = 1/(mT_{EDD})$ .  $T_{EDD}$  – суммарное запаздывание при EDD-расписании;

## 7. RETURN

$\pi^* := (\mathbf{ProcL}(N', t'), j^*, \mathbf{ProcL}(N'', t''))$ , где  
 $N' := \{j_1, \dots, j_k\} \setminus \{j^*\}$ ,  $t' := t$ ,  
 $N'' := \{j_{k+1}, \dots, j_n\}$ ,  $t'' := t + \sum_{i=1}^k p_{j_i}$ .

После каждой итерации "глобальный след"  $\tau_{ij}$  корректируется по правилу

$$\tau_{ij} = (1 - \rho)\tau_{ij} + \rho/T^*,$$

если в "лучшем" найденном расписании на позиции  $i$  обслуживается требование  $j$ . Иначе

$$\tau_{ij} = (1 - \rho)\tau_{ij},$$

где  $\rho \in [0, 1]$  – параметр алгоритма. Значение  $T^*$  – суммарное запаздывание для "лучшего" найденного расписания.

После каждой итерации запускается процедура локального поиска – попарная перестановка требований.

Не трудно убедиться, что трудоемкость алгоритма без локального поиска  $O(mn^2)$ . Процедура ProcL запускается  $n$  раз. В каждой процедуре выполняется порядка  $O(n)$  действий.

Трудоемкость локального поиска  $O(n^3)$ . Тогда трудоемкость **гибридного алгоритма** с локальным поиском составляет не меньше  $O(mn^3)$  операций. То есть теоретическая трудоемкость исследуемых алгоритмов идентична.

### 7.3. Эффективность алгоритмов для тестовых примеров [9]

Вычислительные эксперименты, представленные в этом параграфе, проводились для примеров размерности  $n = 4, \dots, 70, 100$ , которые генерировались с помощью схемы Поттса и Ван Васенхова [9].

Примеры генерировались следующим образом. Значения  $p_j \in Z$  распределены по равномерному закону на промежутке  $[1, 100]$ . Значения  $d_j$  генерировались по равномерному закону на промежутке

$$\left[ \sum_{j=1}^n p_j(1 - TF - RDD/2), \sum_{j=1}^n p_j(1 - TF + RDD/2) \right].$$

Параметры  $TF$  и  $RDD$  принимают значения из множества  $\{0.2, 0.4, 0.6, 0.8, 1\}$  [7, 8, 14]. Для каждой комбинации  $(TF, RDD)$  генерировалось по 100 примеров (всего 2500 примеров для каждого  $n$ ).

Примеры, для которых  $F(\pi_{EDD}) = 0$ , не рассматривались, так как в этом случае **гибридный алгоритм** и **АСО** гарантированно находят точные решения.  $\pi_{EDD}$ -расписание построенное по правилу EDD.

В алгоритмах использовались следующие параметры:  $\alpha = 1; \beta = 2; \rho = 0,1$ . Локальный поиск – попарная перестановка. Применяемая эвристика – MDD [14].

Для каждого примера запускался точный **алгоритм А**. Мы получали точное решение  $F_{opt}$ .

В **алгоритме АСО** муравьи генерировались до тех пор, пока не находилось расписание  $\pi$ , при котором  $F(\pi) = F_{opt}$ . Это "необходимое" количество муравьев фиксировалось. Количество муравьев нами было ограничено  $m \leq 100$ .

Алгоритм запускался до 10 раз (пока не найдено оптимальное расписание). Таким образом мы пытались избежать "случайности" работы алгоритма.

Лучше найденное решение  $F_{ACO}$  фиксировалось. Вычислялась относительная погрешность  $\frac{F_{ACO}-F_{opt}}{F_{opt}}$ .

Такая же схема использовалась для **Гибридного алгоритма**. Таким образом мы могли сравнить относительную погрешность двух алгоритмов (**ACO** и **Гибридного**), количество муравьев, необходимое каждому алгоритму для поиска оптимального расписания.

Результаты экспериментов представлены в таблице 1.

Таблица 1. Результаты экспериментов на примерах Поттса и Ван Вассенхова.

n	Кол. прим.	не опт. АСО	не опт. Гиб.	rel. АСО	rel. Гиб.	Мур. АСО	Мур. Гиб.
4	2500	0	0	0	0	1.0404	1.004
5	2500	0	0	0	0	1.0616	1.0216
6	2500	0	0	0	0	1.0908	1.036
7	2500	0	0	0	0	1.1384	1.0612
8	2500	0	0	0	0	1.1968	1.0588
9	2500	0	0	0	0	1.1456	1.1044
10	2500	0	0	0	0	1.2576	1.1228
11	2500	0	0	0	0	1.2364	1.126
12	2500	0	0	0	0	1.2672	1.1484
13	2500	0	0	0	0	1.2968	1.2296
14	2500	0	0	0	0	1.3704	1.2464
15	2500	0	0	0	0	1.3576	1.2744
16	2500	0	0	0	0	1.4324	1.3928
17	2500	0	0	0	0	1.4376	1.3196
18	2500	0	0	0	0	1.4656	1.3216
19	2500	2	0	0.22	0	1.6164	1.4004
20	2500	1	0	0.58	0	1.6064	1.4204
21	2500	0	0	0	0	1.5892	1.4232
22	2500	1	0	0.16	0	1.626	1.4844
23	2500	0	0	0	0	1.6572	1.5184
24	2500	0	0	0	0	1.6992	1.5568
25	2500	0	0	0	0	1.8128	1.5504
26	2500	0	0	0	0	1.7736	1.584
27	2500	0	0	0	0	1.88	1.6828
28	2500	2	0	0.09	0	1.9704	1.6688
29	2500	0	0	0	0	1.9172	1.7872
30	2500	0	0	0	0	1.9744	1.7268
31	2500	0	0	0	0	1.9656	1.8656

Продолжение табл.1

n	Кол. прим.	не опт. АСО	не опт. Гиб.	rel. АСО	rel. Гиб.	Мур. АСО	Мур. Гиб.
32	2500	0	0	0	0	2.1688	1.8788
33	2500	0	0	0	0	2.214	1.844
34	2500	1	0	0.15	0	2.2212	1.9568
35	2500	0	0	0	0	2.3272	2.1152
36	2500	0	1	0	0.04	2.2332	2.154
37	2500	1	1	0.38	0.01	2.4796	2.102
38	2500	0	0	0	0	2.2696	2.1172
39	2500	0	0	0	0	2.576	2.1044
40	2500	1	0	0.04	0	2.6036	2.2424
41	2500	0	0	0	0	2.552	2.2704
42	2500	1	1	0.05	0.01	2.7888	2.4092
43	2500	1	1	0.07	0.06	2.7316	2.3656
44	2500	3	0	0.04	0	2.8464	2.3784
45	2500	2	0	0.68	0	2.9736	2.4728
46	2500	1	0	0.03	0	3.1624	2.4088
47	2500	2	0	0.01	0	3.248	2.5152
48	2500	9	0	0.56	0	3.4516	2.5196
49	2500	3	1	0.15	0.08	3.4252	2.7
50	2500	9	1	0.35	0.29	3.716	2.6336
51	2500	8	0	0.22	0	3.8412	2.7768
52	2500	4	1	0.04	0.07	3.5816	2.86
53	2500	4	2	0.03	0.42	3.8948	2.9668
54	2500	9	3	0.1	0.29	4.0324	2.9924
55	2500	8	2	0.11	0.06	4.1048	3.0496
56	2500	9	1	0.83	0.01	4.2916	3.0064
57	2500	7	0	0.23	0	4.1568	3.158
58	2500	14	0	0.17	0	4.71	3.3724
59	2500	14	4	0.24	0.1	4.81	3.3372

Окончание табл.1

n	Кол. прим.	не опт. АСО	не опт. Гиб.	rel. АСО	rel. Гиб.	Мур. АСО	Мур. Гиб.
60	2500	11	1	0.22	0.01	4.7268	3.4224
61	2500	18	2	1.26	0.02	5.3032	3.5216
62	2500	10	2	0.26	0.01	5.0964	3.5032
63	2500	17	7	0.16	0.08	5.3016	3.5728
64	2500	15	6	0.57	0.46	5.2388	3.6504
65	2500	18	7	0.1	0.14	5.548	3.6604
66	2500	17	11	0.15	0.14	5.4288	3.8552
67	2500	17	7	0.83	0.1	6.1068	4.1016
68	2500	25	4	0.2	0.08	6.3864	3.7252
69	2500	18	6	0.12	0.1	6.1912	4.0796
70	2500	33	4	0.23	0.05	6.974	3.8672
100	617	36	0	0.31	0	27.35	4.66

В первой колонке представлено значение  $n$  – размерность задачи. Во второй колонке – количество рассмотренных примеров. В третьей и четвертой колонках, соответственно, количество примеров, для которых **алгоритм АСО** или **гибридный** не нашли точные решения. В пятой и шестых колонках представлена максимальная относительная погрешность алгоритмов (с точностью до сотых процента). В последних двух колонках – среднее количество муравьев необходимое для поиска оптимального решения.

Как видно из таблицы, примерно для 99% примеров оба алгоритма находят точные решения.

Количество примеров, неточно решенных **гибридным алгоритмом**, значительно меньше, чем в **алгоритме АСО**, и не превосходит 0.44% от общего числа примеров. Относительная погрешность не превосходит 0.46%. Количество муравьев, необходимое **гибридному алгоритму** также меньше.

Относительная погрешность **алгоритма АСО** превосходит 1.26% для  $n = 61$ . Количество "неточно" решенных примеров для  $n = 70$  больше 1%.

Следует ожидать, что с ростом  $n$  будет наблюдаться значительное преимущество **гибридного алгоритма**.

#### 7.4. Эффективность алгоритмов для случая В-1

В этом параграфе представлены экспериментальные исследования алгоритмов для примеров случая В-1 2.

Эксперименты, аналогичные экспериментам из предыдущего параграфа, проводились для примеров размерности  $n = 4, \dots, 100$ . Для каждого значения  $n$  рассматривалось по 1000 примеров случая В-1.

Значения  $p_j$  генерировались по равномерному закону на промежутке  $[1, 500]$ . Директивные сроки  $d_j$  распределены равномерно на интервале  $[A, A + p_n]$ , где  $A \in [0, \sum p_j - p_n]$ .

Схема проведения экспериментов и параметры алгоритмов описаны в предыдущем параграфе. В качестве точного алгоритма использовался **алгоритм В-1 модифицированный**, трудоемкость которого не больше  $O(n \sum p_j)$  для целочисленных примеров.

Были получены следующие результаты.

Таблица 2. Результаты экспериментов на примерах В-1.

n	Кол. прим.	не опт. АСО	не опт. Гиб.	rel. АСО	rel. Гиб.	Мур. АСО	Мур. Гиб.
4	1000	0	0	0	0	1.038	1.005
5	1000	0	0	0	0	1.071	1.034
6	1000	0	0	0	0	1.159	1.073
7	1000	1	0	0.67	0	1.38	1.044
8	1000	0	0	0	0	1.311	1.058
9	1000	0	0	0	0	1.401	1.137
10	1000	0	0	0	0	1.418	1.088
11	1000	0	0	0	0	1.584	1.091
12	1000	0	0	0	0	1.478	1.247
13	1000	0	0	0	0	1.495	1.201
14	1000	0	0	0	0	1.441	1.207
15	1000	0	0	0	0	1.621	1.241
16	1000	0	0	0	0	1.501	1.279
17	1000	0	0	0	0	1.45	1.271
18	1000	0	0	0	0	1.526	1.805
19	1000	0	0	0	0	1.511	1.36
20	1000	0	0	0	0	1.448	1.252
21	1000	0	0	0	0	1.457	1.457
22	1000	0	0	0	0	1.448	1.373
23	1000	0	1	0	0	1.481	1.761
24	1000	0	0	0	0	1.446	1.644
25	1000	0	2	0	0	1.337	1.696
26	1000	0	3	0	0.01	1.381	1.871
27	1000	0	1	0	0.01	1.429	1.707
28	1000	0	1	0	0	1.532	1.8
29	1000	0	3	0	0	1.423	1.815
30	1000	0	2	0	0	1.311	2.027
31	1000	0	4	0	0.01	1.354	1.929

Продолжение табл. 2

n	Кол. прим.	не опт. АСО	не опт. Гиб.	rel. АСО	rel. Гиб.	Мур. АСО	Мур. Гиб.
32	1000	0	2	0	0	1.343	1.98
33	1000	0	3	0	0	1.379	2.005
34	1000	0	2	0	0	1.166	1.764
35	1000	0	7	0	0	1.287	2.435
36	1000	0	4	0	0	1.288	1.894
37	1000	0	5	0	0	1.237	2.102
38	1000	0	5	0	0	1.266	2.027
39	1000	0	5	0	0	1.216	2.115
40	1000	0	4	0	0	1.187	2.043
41	1000	0	3	0	0	1.241	1.868
42	1000	0	6	0	0	1.212	2.162
43	1000	0	4	0	0	1.287	2.042
44	1000	0	4	0	0	1.335	1.861
45	1000	0	2	0	0	1.304	2.149
46	1000	0	3	0	0	1.239	1.895
47	1000	0	4	0	0	1.224	1.847
48	1000	0	5	0	0	1.251	2.298
49	1000	0	3	0	0	1.264	2.179
50	1000	0	1	0	0	1.168	1.712
51	1000	0	0	0	0	1.251	1.332
52	1000	0	5	0	0	1.22	1.82
53	1000	0	6	0	0	1.213	1.995
54	1000	0	2	0	0	1.189	1.59
55	1000	0	1	0	0	1.139	1.639
56	1000	0	5	0	0	1.107	2.075
57	1000	0	5	0	0	1.18	2.049
58	1000	0	4	0	0	1.208	2.175
59	1000	0	0	0	0	1.218	1.424
60	1000	0	5	0	0	1.114	2.076

Продолжение табл. 2

n	Кол. прим.	не опт. АСО	не опт. Гиб.	rel. АСО	rel. Гиб.	Мур. АСО	Мур. Гиб.
61	1000	0	4	0	0	1.15	1.773
62	1000	0	6	0	0	1.123	2.154
63	1000	0	4	0	0	1.114	1.909
64	1000	0	0	0	0	1.137	1.207
65	1000	0	4	0	0	1.112	1.854
66	1000	0	4	0	0	1.237	1.798
67	1000	0	1	0	0	1.132	1.57
68	1000	0	1	0	0	1.098	1.412
69	1000	0	4	0	0	1.12	1.912
70	1000	0	6	0	0	1.076	1.904
71	1000	0	5	0	0	1.105	1.907
72	1000	0	4	0	0	1.123	1.765
73	1000	0	3	0	0	1.084	1.589
74	1000	0	2	0	0	1.11	1.527
75	1000	0	4	0	0	1.122	1.656
76	1000	0	4	0	0	1.122	1.688
77	1000	0	0	0	0	1.177	1.382
78	1000	0	2	0	0	1.088	1.532
79	1000	0	6	0	0	1.122	2.114
80	1000	0	6	0	0	1.104	1.97
81	1000	0	3	0	0	1.103	1.553
82	1000	0	2	0	0	1.103	1.602
83	1000	0	2	0	0	1.18	1.653
84	1000	0	2	0	0	1.08	1.603
85	1000	0	3	0	0	1.111	1.555
86	1000	0	1	0	0	1.149	1.534
87	1000	0	0	0	0	1.11	1.415
88	1000	0	2	0	0	1.123	1.401

Окончание табл. 2

n	Кол. прим.	не опт. АСО	не опт. Гиб.	rel. АСО	rel. Гиб.	Мур. АСО	Мур. Гиб.
89	1000	0	1	0	0	1.087	1.484
90	1000	0	3	0	0	1.086	1.596
91	1000	0	4	0	0	1.083	1.76
92	1000	0	4	0	0	1.094	1.936
93	1000	0	2	0	0	1.097	1.519
94	1000	0	2	0	0	1.096	1.463
95	1000	0	2	0	0	1.093	1.459
96	1000	0	5	0	0	1.095	1.963
97	1000	0	0	0	0	1.079	1.134
98	1000	0	0	0	0	1.125	1.238
99	1000	0	1	0	0	1.073	1.275
100	1000	0	1	0	0	1.068	1.525

Необходимо заметить, что мы вычисляли относительную погрешность с точностью до сотых процента. Поэтому в таблице результатов можно наблюдать ситуацию, когда количество неточно решенных примеров больше 0, а относительная погрешность равна 0 (например для  $n = 23$ ).

**Алгоритм АСО** находит точное решение практически для всех примеров. Единственный неточно решенный пример встретился для  $n = 7$ . **Гибридный алгоритм** находит точное решение для 99% примеров. Причем относительная погрешность **гибридного алгоритма** не превосходит 0,01%.

В среднем обоим алгоритмам требуется не более двух муравьев чтобы найти точное решение.

Как видно из табл. 2 эффективность **гибридного алгоритма**, использующего идеи **алгоритма А**, ниже эффективности **алгоритма АСО** на примерах В-1. Можно объяснить это тем, что примеры В-1 "наиболее сложные" для **алгоритма А**.

Заметим также, что при такой генерации примеров, трудоемкость точного **алгоритма В-1 модифицированный** меньше  $O(n^3)$ , так как генерацией "не захватывается" NP-трудный подслучай.

### 7.5. Эффективность алгоритмов для канонических DL-примеров [5]

Мы генерировали примеры ЧНР для  $n = 4, \dots, 40$ . Параметры  $\delta_i$  распределены по равномерному закону на промежутке  $[1, 50]$ . Пример ЧНР задается следующим образом  $b_{2n} := 1$ ,  $b_{2n-1} := b_{2n} + \delta_n$ ,  $b_{2i} := b_{2i+1} + 1$ ,  $b_{2i-1} := b_{2i} + \delta_i$ ,  $i := 1, \dots, n-1$ .

За полиномиальное время мы преобразовывали пример ЧНР к каноническому DL-примеру (количество требований  $3n + 1 = 13, 16, \dots, 121$ ). Для канонического DL-примера запускался точный псевдополиномиальный **алгоритм В-1 канонический**, трудоемкостью  $O(n\delta)$ , **алгоритмы АСО** и **Гибридный**. Параметры алгоритмов прежние.

Для каждого значения  $n$  генерировалось по 50 примеров. Каждый алгоритм (**гибридный** и **АСО**) запускался 1 раз.

Были получены следующие результаты.

Таблица 3. Результаты экспериментов на канонических DL-примерах.

$3n + 1$	Кол. прим.	не опт. АСО	не опт. Гиб.	rel. АСО	rel. Гиб.	Мур. АСО	Мур. Гиб.
13	50	0	0	0	0	1.96	1.94
16	50	1	0	0	0	4.92	3.4
19	50	2	3	0	0	10.3	11.64
22	50	2	2	0	0	7.66	8.22
25	50	4	4	0	0	16.28	16.44
28	50	6	4	0	0	19.2	15.24

$3n + 1$	Кол. прим.	не опт. АСО	не опт. Гиб.	rel. АСО	rel. Гиб.	Мур. АСО	Мур. Гиб.
31	50	0	3	0	0	8.16	15.66
34	50	1	1	0	0	8.8	9.44
37	50	1	0	0	0	7.12	7.58
40	50	0	0	0	0	5.88	4.74
43	50	0	0	0	0	4.14	5.76
46	50	0	0	0	0	3.32	4.48
49	50	0	0	0	0	4.52	4.76
52	50	0	0	0	0	3.28	4.48
55	50	0	0	0	0	3.36	4.26
58	50	0	0	0	0	3.82	4.58
61	50	0	0	0	0	3.04	4.36
64	50	0	0	0	0	3.48	3.46
67	50	0	0	0	0	3.22	3.48
70	50	0	0	0	0	2.26	3.1
73	50	0	0	0	0	2.46	3.36
76	50	0	0	0	0	2.96	3.22
79	50	0	0	0	0	2.22	2.52
82	50	0	0	0	0	2.94	3.24
85	50	0	0	0	0	3.34	3.72
88	50	0	0	0	0	2.8	3.56
91	50	0	0	0	0	2.64	2.6
94	50	0	0	0	0	2.7	2.8
97	50	0	0	0	0	2.7	2.9
100	50	0	0	0	0	2.46	2.68
103	50	0	0	0	0	2.48	2.52
106	50	0	0	0	0	3.08	2.46
109	50	0	0	0	0	2.44	2.2
112	50	0	0	0	0	2.18	2.22
115	50	0	0	0	0	2.08	2.12
118	50	0	0	0	0	2.02	1.96
121	50	0	0	0	0	2.18	2.56

Эффективность алгоритмов примерно идентична. Только для  $3n + 1 = 25, 28$  количество неточно решенных примеров достигает 10%. При этом погрешность алгоритмов меньше 0.01%. Среднее необходимое количество итераций для нахождения точного решения не превышает 20.

Стоит отметить, что эти метаэвристические алгоритмы находят точное решение для канонических DL-примеров размерности  $3n + 1 = 121$ . Для решения таких примеров необходимо перебрать порядка  $2^n$  канонических DL-расписаний[5]. В **гибридном алгоритме** для каждой пары требований  $V_{2i-1}, V_{2i}$ ,  $i := n, \dots, 1$  рассматривается только 2 порядка обслуживания: требование  $V_{2i-1}$  обслуживается на позиции  $2i - 1$ , а требование  $V_{2i}$  обслуживается на позиции  $3n + 1 - (i - 1)$  и наоборот. Причем вероятности постановки в эти позиции примерно равны  $1/2$  на каждой итерации. То есть позиции "равновероятны". Следовательно, вероятность найти точное решение стремится к величине  $o(1/2^n)$ .

Мы повторили эксперимент на тех же примерах, при тех же условиях, отключив в алгоритмах "локальный поиск". Результаты приводятся в табл.4:

Таб.4. Результаты экспериментов без локального поиска.

$3n + 1$	Кол. прим.	не опт. АСО	не опт. Гиб.	rel. АСО	rel. Гиб.	Мур. АСО	Мур. Гиб.
13	50	50	26	13.46	0.01	100	58.2
16	50	50	35	0.77	0.03	100	73.82
19	50	50	43	3.29	2.78	100	88.06
22	50	50	46	2.86	2.05	100	93.52
25	50	50	49	2.03	1.58	100	98.02
28	50	50	50	2.97	2.49	100	100
31	50	50	49	3.48	2.02	100	98.48

$3n + 1$	Кол. прим.	не опт. АСО	не опт. Гиб.	rel. АСО	rel. Гиб.	Мур. АСО	Мур. Гиб.
34	50	50	49	2.85	1.67	100	98.4
37	50	50	49	1.71	1.41	100	98.02
40	50	50	50	0.96	1.79	100	100
43	50	50	50	2.3	2.06	100	100
46	50	50	50	1.16	2.24	100	100
49	50	50	50	2.18	2.36	100	100
52	50	50	50	1.61	1.75	100	100
55	50	50	50	1.42	1.87	100	100
58	50	50	50	1.08	1.4	100	100
61	50	50	50	1.22	1.51	100	100
64	50	50	50	1.37	1.6	100	100
67	50	50	50	2.41	1.66	100	100
70	50	50	50	1.82	1.71	100	100
73	50	50	50	1.52	1.57	100	100
76	50	50	50	1.71	1.61	100	100
79	50	50	50	2.47	1.49	100	100
82	50	50	50	2.44	1.65	100	100
85	50	50	50	2.02	2.69	100	100
88	50	50	50	1.93	2.03	100	100
91	50	50	50	2.79	1.79	100	100
94	50	50	50	2.28	1.46	100	100
97	50	50	50	1.95	1.37	100	100
100	50	50	50	2.21	1.75	100	100
103	50	50	50	1.48	1.74	100	100
106	50	50	50	2.05	1.56	100	100
109	50	50	50	1.78	1.4	100	100
112	50	50	50	1.97	1.76	100	100
115	50	50	50	1.76	1.95	100	100
118	50	50	50	1.79	1.98	100	100
121	50	50	50	2.27	1.38	100	100

Анализируя обе таблицы можно сделать вывод, что эффективность алгоритмов на данном классе примеров достигается в основном за счет локального поиска. Причем количество запусков локального поиска на каждой итерации может быть экспоненциально.

При  $3n + 1 = 40$  оба алгоритма неточно решают все примеры.

## 8. Экспериментальные исследования алгоритмов решения задачи

Были проведены многочисленные эксперименты для выяснения количества *точек изменения расписания* для *канонических DL-примеров*. Результаты экспериментов позволили оценить трудоемкость **алгоритма**

**В-1 модифицированный**. Результаты проведенных экспериментов (№№ 6, 8, 8.2) представлены в сводной таблице после эксперимента №8.

В ходе экспериментов последовательно перебирались примеры **ЧНР** из некоторого класса. Каждый пример за полиномиальное число шагов (от  $n$ ) приводился к каноническому DL-примеру задачи  $1 || \sum T_j$ . Для канонического DL-примера запускался **алгоритм В-1 канонический**, на каждом шаге  $l$  которого подсчитывалось количество *точек  $t$  изменения расписания*.

**Эксперимент №6.** Перебирались все примеры **ЧНР** с количеством групп  $n = 2, 3, 4, 5$  для которых выполняется:

$$\begin{cases} 200 \geq b_1 > b_2 \geq b_3 > b_4 \geq \dots \geq b_{2n-1} > b_{2n} \geq 1, \\ b_i \in Z^+. \end{cases}$$

Ограничение  $200 \geq b_1$  обусловлено вычислительными мощностями компьютера. Так как перебираются все примеры  $200 \geq b_1 > b_2 \geq b_3 > b_4 \geq \dots \geq b_{2n-1} > b_{2n}$ , то их число превосходит количество сочетаний  $C_{200}^{2*n}$ . Для размерности  $n = 5$  число примеров превосходит:

$$C_{200}^{10} = \frac{200!}{(200-10)!10!} = 22'451'004'309'013'280$$

Увеличение  $b_1$  значительно увеличит количество рассматриваемых примеров, что в разумные сроки не могло быть проанализировано.

Получены следующие результаты:

п (количество групп)	Макс. кол-во точек изм. расписания
2	1
3	4
4	11
5	21

Количество точек изменения расписания считается на каждом шаге  $l = n, n - 1, \dots, 1$  и суммируется.

Детальный анализ результатов позволил выявить следующее важное свойство:

**Свойство 1.** Два примера ЧНР

$\{(a_1, a_2), (a_3, a_4), \dots, (a_{2j-1}, a_{2j}), \dots, (a_{2n-1}, a_{2n})\}$   
и  $\{(a_1, a_2), (a_3, a_4), \dots, (a_{2j-1} + \Delta, a_{2j} + \Delta), \dots, (a_{2n-1}, a_{2n})\}$ ,  
где  $\Delta \in Z^+$ ,  $a_{2j-1} + \Delta \leq a_{2j-2}$ ,  $a_{2j} + \Delta \leq a_{2j+1}$   
(примеры с одинаковым распределением  $\delta_i$ ) эквивалентны.  
Соответствующие канонические примеры имеют одинаковое количество точек изменения расписания и одинаковое оптимальное расписание.

Учитывая данное свойство, примеры ЧНР будем обозначать через  $(\delta_1, \delta_2, \dots, \delta_n)$ .

**Эксперимент №8.** Перебирались все примеры ЧНР  $(\delta_1, \delta_2, \dots, \delta_n)$ . В полной мере эксперимент завершен для примеров с количеством групп  $n = 2, 3, \dots, 7$  для которых выполняется:

$$\begin{cases} 50 \geq \sum_{i=1}^n \delta_i, \\ \delta_i \in Z^+. \end{cases}$$

Обозначим  $CS_l$  – количество точек изменения расписания на шаге  $l = n, n - 1, \dots, 1$  алгоритма В-1 канонический.

Были найдены сложные примеры ЧНР, для которых  $CS_n = 1, CS_{n-1} = 3, CS_{n-2} \geq 7, CS_{n-3} \geq 15, CS_{n-4} \geq 31, \dots$ . Выявлены следующие свойства:

**Свойство 2.** Пусть  $S$  - множество примеров, для которых  $\delta_1 = \sum_{i=2}^n \delta_i$  (\*). Пусть  $U$  - множество сложных примеров. То примеры (\*) в своем большинстве принадлежат примерам с повышенной сложностью,  $S \cap U \neq \emptyset$ .

**Свойство 3.** Среди примеров с повышенной сложностью, обладающих свойством (\*), около 70% имеют сложность (1,3,7,15,31,0) (для размерности  $n = 6$ ). Можно сделать предположения, что сложность для произвольного  $n$  будет иметь вид  $(1, 3, 7, \dots, a_i, 2a_i + 1, \dots)$ .

**Свойство 4.** Если из сложного примера размерности  $n$  "убрать" первую группу, то полученный пример размерности  $n - 1$  не обязательно сложный.

**Свойство 5.** Пусть пример  $(\delta_1, \delta_2, \dots, \delta_n)$  имеет сложность (1,3,7,15,31), то такую же сложность имеет пример  $(\delta_1 + k * 2, \delta_2, \dots, \delta_n)$ .

**Свойство 6.** К классу сложных примеров относятся не только "разрешимые" примеры **ЧНР** (т.е. решение примера "НЕТ").

**Свойство 7.** Очевидно, если перенумеровать пары чисел задачи **ЧНР** таким образом, что  $\delta_1 \leq \delta_2 \leq \dots \leq \delta_n$ , то получим эквивалентный пример. Но при этом снижается трудоемкость **алгоритма В-1 канонический** и часто уменьшается количество *точек изменения расписания*. Для такой нумерации групп проведен аналогичный эксперимент №8.2. Стоит отметить, что примеры  $SD$  удовлетворяют такой нумерации.

Получены следующие результаты:

n	B-1	B-1 канонический $(CS_n, CS_{n-1}, \dots, CS_1)$	
		$\delta_1, \delta_2, \dots, \delta_n$	$\delta_1 \leq \delta_2 \leq \dots \leq \delta_n$
2	1	(1,0)	(1,0)
3	3	(1,3,0)	(1,1,0)
4	8	(1,3,8,0)	(1,3,2,0)
5	15	(1,3,8,18,0)	(1,3,5,5,0)
6	23	(1,3,8,18,32,0)	(1,3,7,7,7,0)
7	38	(1,3,8,18,32,63,0)	(1,3,7,13,19,19,0)
8	44	-	-
9	51	-	-

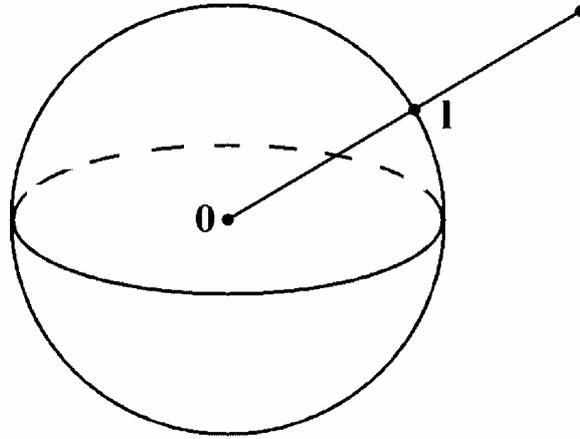


Рис. 19. Множество примеров в  $2n$ -мерном пространстве

## 9. Методика проведения экспериментов

Для исследования экспериментальной эффективности алгоритмов решения задач теории расписаний нам бы хотелось предложить свою схему проведения экспериментов. Данная схема не ограничивается только задачей минимизации суммарного запаздывания.

Параметры требований можно представить как вектор в  $2n$ -мерном пространстве  $(p_1, p_2, \dots, p_n, d_1, d_2, \dots, d_n)$ . Как было показано выше мы всегда можем рассматривать задачу  $1||\sum T_j$  с момента освобождения прибора  $t_0$ . Очевидно, что примеры  $\langle \{p_j, d_j\}_{j \in N}, 0 \rangle$  и  $\langle \{\alpha p_j, \alpha d_j\}_{j \in N}, 0 \rangle$ ,  $\alpha > 0$ , эквивалентны.

Все примеры, расположенные на луче, выходящем из 0, являются эквивалентными. Следовательно, можно

рассматривать только точки (примеры) расположенные на единичной сфере в  $2n$ -мерном пространстве.

Эксперименты проводились следующим образом. Выбиралась произвольная точка  $x_0$  на сфере (если  $p_j < 0$ , то умножается на  $-1$ ). В окрестности этой точки с небольшим радиусом ( $\varepsilon \leq 0.1$ ) выбирались точки случайным образом (по равномерному распределению для каждого из параметров). В каждой точке искалось оптимальное расписание с помощью **алгоритма А**, оценивалось количество ветвлений в дереве поиска. В  $\varepsilon$ -окрестности проводилось до миллиона экспериментов и выбиралась точка  $x_1$  с наибольшим количеством ветвлений  $\mu(x_1)$ . Если  $\mu(x_1) > \mu(x_0)$ , то "переходим" в точку  $x_1$  и в  $\varepsilon$ -окрестности вновь "ищем" более сложную точку  $x_2$  и т.д. Останавливали наш процесс поиска, когда в  $\varepsilon$ -окрестности не удавалось найти "более сложную точку". Таким образом, была выделена совокупность "сложных" примеров относительно **алгоритма А**. На рис.20 показана окрестность "около сложной точки". Замечен интересный факт: "сложные" точки "почти" ортогональны между собой.

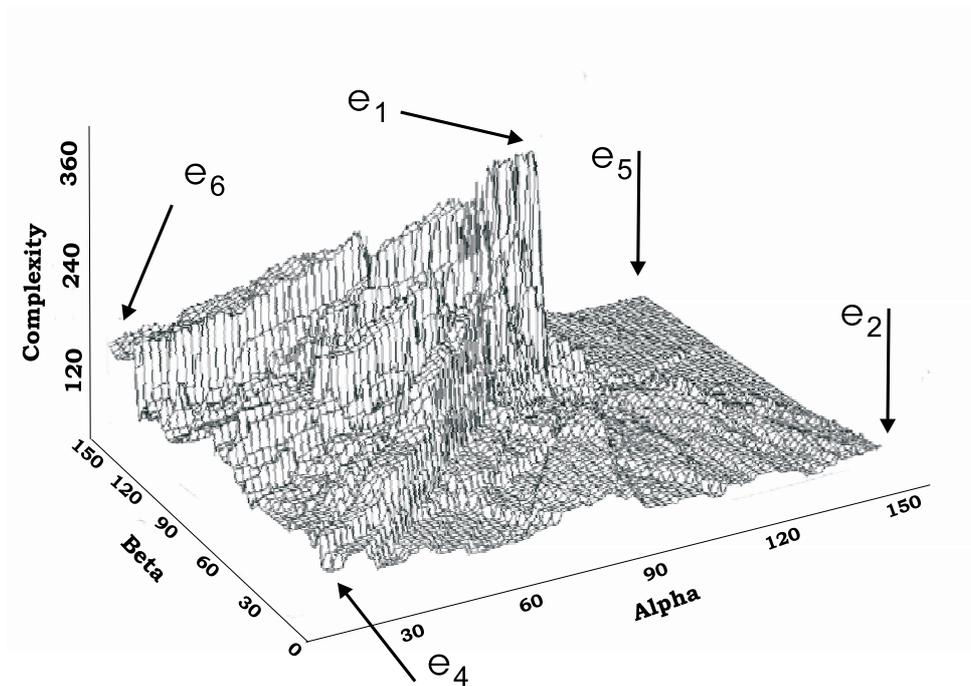


Рис. 20. Окрестность "сложной точки"

## 10. Заключение

Для частного случая проблемы предложены псевдополиномиальные и полиномиальные алгоритмы. Для NP-трудного случая **В-1** приводятся новые подходы к решению. Проведены экспериментальные исследования предложенных алгоритмов.

Показано, что алгоритмы, использующие только следующие правила исключения: "правила исключения 1-4, использование  $E_j$  и  $L_j$ , построение модифицированного примера", имеют экспоненциальную трудоемкость. Вызывает сомнение, что такими алгоритмами можно решать примеры размерностью  $n \geq 100$ .

Предложен новый **гибридный алгоритм**, который, по результатам экспериментов, существенно эффективнее **алгоритма АСО** для тестовых примеров Поттса и Ван Васенхова [9].

Для NP-трудных канонических DL-примеров "хорошая эффективность" **гибридного алгоритма** и **АСО** достигается за счет локального поиска.

При оценке эффективности алгоритмов для задачи 1|| $\sum T_j$  не достаточно использовать тестовые примеры из работы [9].

## Список литературы

1. Танаев В.С., Шкурба В.В. Введение в теорию расписаний. М: Наука, 1975.
2. Танаев В.С., Гордон В.С., Шафранский Я.М. Теория расписаний. Одностадийные системы. М: Наука, 1984.
3. Танаев В.С., Сотсков Ю.Н., Струевич В.А. Теория расписаний. Многостадийные системы. М: Наука, 1989.
4. Танаев В.С., Ковалёв М.Я., Шафранский Я.М. Теория расписаний. Групповые технологии. Минск: Институт технической кибернетики, 1998.
5. J. Du and J. Y.-T. Leung, *Minimizing total tardiness on one processor is NP-hard* // Math. Oper. Res.. 1990. № 15. P. 483–495.
6. E.L. Lawler, *A pseudopolynomial algorithm for sequencing jobs to minimize total tardiness* // Ann. Discrete Math.. 1977. № 1. P. 331–342.
7. W. Szwarc, F. Della Croce and A. Grosso, *Solution of the single machine total tardiness problem* // Journal of Scheduling. 1999. № 2. P. 55–71.
8. W. Szwarc, A. Grosso and F. Della Croce, *Algorithmic paradoxes of the single machine total tardiness problem* // Journal of Scheduling. 2001. № 4. P. 93-104.
9. C.N. Potts and L.N. Van Wassenhove, *A decomposition algorithm for the single machine total tardiness problem* // Oper. Res. Lett.. 1982. № 1. P. 177–182.

10. F. Della Croce, A. Grosso, V. Paschos, *Lower bounds on the approximation ratios of leading heuristics for the single-machine total tardiness problem* // Journal of Scheduling. 2004. № 7. P. 85–91
11. А.А. Лазарев, *Эффективные алгоритмы решения некоторых задач теории расписаний для одного прибора с директивными сроками обслуживания требований*. Диссертация на соискание ученой степени канд. физ.-матем. наук Казань. 1989. 108 с.
12. A. Lazarev, A. Kvaratskhelia, A. Tchernykh, *Solution algorithms for the total tardiness scheduling problem on a single machine* // Workshop Proceedings of the ENC'04 International Conference. 2004. P. 474–480.
13. S. Chang, Q. Lu, G. Tang, W. Yu, *On decomposition of total tardiness problem* // Oper. Res. Lett.. 1995. № 17. P. 221–229.
14. A. Bauer, B. Bullnheimer, R.F.Hartl, C. Strauss. *Minimizing Total Tardiness on a Single Machine Using Ant Colony Optimization*. // Proceedings of the 1999 Congress on Evolutionary Computation (CEC99), 6-9 July Washington D.C., USA. 1999. P. 1445–1450.
15. D. Merkle, M. Middendorf. *An Ant Algorithm with a New Pheromone Evaluation Rule for Total Tardiness Problem*. // EvoWorkShops 2000, LNCS 1803, Springer-Verlag. 2000. P. 287–296.
16. H. Emmons. *One machine sequencing to minimize certain functions of job tardiness* // Oper. Res., 17. 1969. P. 701–715.