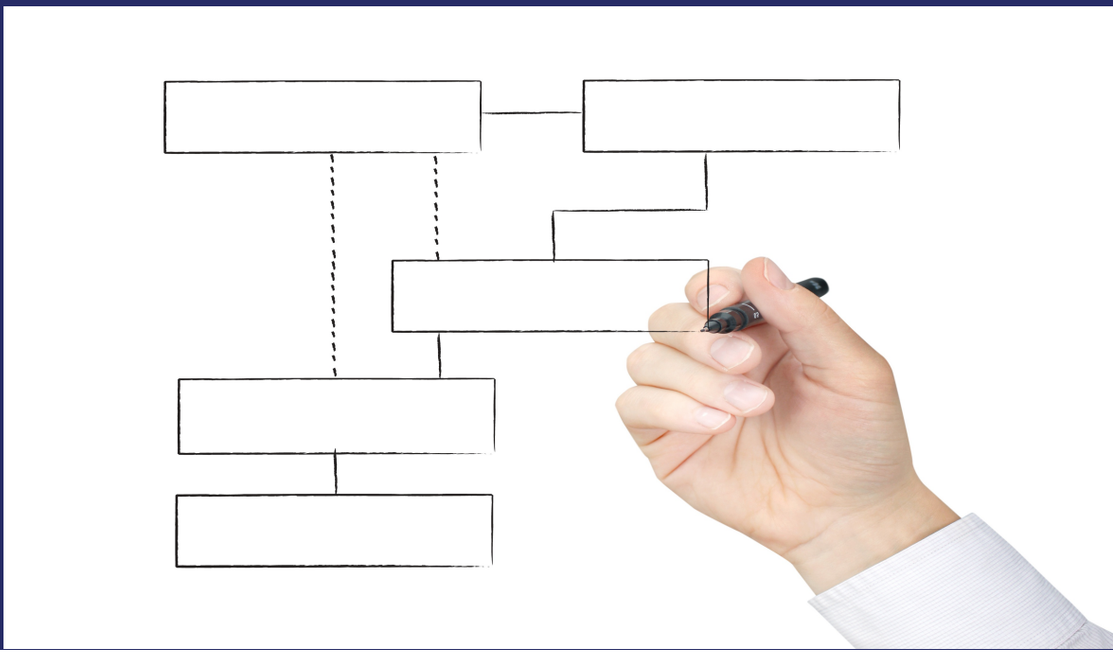
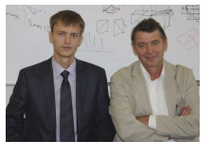
Фундаментальными задачами теории расписаний для одного прибора являются задачи с критериями минимизации суммарного запаздывания и задачи минимизации максимального временного смещения. В данной книге приводится достаточно полное исследование NP-трудной в обычном смысле задачи минимизации суммарного запаздывания (total tardiness) и ее взаимосвязь с задачей Разбиения. Выделен ряд новых полиномиально и псевдо-полиномиальных разрешимых случаев данной задачи. При исследовании были использованы как стандартные методы дискретной оптимизации (метод динамического программирования, - графическая модификация), так и методы, учитывающие специфические особенности задачи. Наряду с точными методами применялись и приближенные метаэвристические подходы (метод "муравьиные колонии"). С помощью графического подхода удалось показать полиномиальную разрешимость обратной задачи - максимизации суммарного запаздывания.





Александр Лазарев

Лазарев Александр Алексеевич, доктор физ.-мат. наук, профессор, заведующий лабораторией Института Проблем Управления РАН. e-mail: Lazarev@ipu.ru Гафаров Евгений Рашидович, кандидат физ.-мат. наук, старший научный сотрудник Института Проблем Управления РАН. e-mail: axel73@mail.ru Научные интересы: задачи оптимизации, теория расписаний, алгоритмы

978-3-8443-5789-9

Александр Лазарев
Евгений Гафаров

ТЕОРИЯ РАСПИСАНИЙ

Задачи суммарного запаздывания для одного прибора

LAP LAMBERT Academic Publishing

А.А. ЛАЗАРЕВ, Е.Р. ГАФАРОВ

ТЕОРИЯ РАСПИСАНИЙ.
ЗАДАЧИ СУММАРНОГО ЗАПАЗДЫВАНИЯ ДЛЯ ОДНОГО ПРИБОРА

Содержание.

Введение.

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 $1 \mid\mid \sum T_j$ 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. Кварацхелия за помощь в работе и обсуждение результатов.

А.А. Лазарев, Е.Р. Гафаров.
Май 2011 г.

Глава 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 $j \in N = \{1, 2, \ldots, n\}$ 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 π строится с момента времени t_0 и однозначно задается перестановкой элементов множества N.

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

Исследуемая задача является NP-трудной в обычном смысле [5]. Е.Л. Лаулер [6] предложил псевдополиномиальный алгоритм решения общего случая задачи трудоёмкости $O(n^4 \sum p_j)$. В. Шварц и др. построили [7, 8] алгоритмы решения задачи, которые были протестированы для примеров $n < 600$ (тестовые примеры С.Н. Поттса и Л.Н. Ван Вассенхова [9]). Исследование приближенных алгоритмов решения задачи было проведено в работе [10], где построены примеры, на которых известные приближенные алгоритмы находят решение с относительной погрешностью порядка размерности примера n.

В главе 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 главе 3. В основе приведенных алгоритмов лежит правило разбиения исходного множества требований на подмножества. Для каждого подмножества рекурсивной процедурой находится частичное оптимальное расписание.

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

В главе 5 показано, что известные алгоритмы [7, 8, 11, 13], для которых не получена оценка трудоёмкости, имеют экспоненциальную трудоёмкость для некоторых частных случаев задачи $1 \mid\mid \sum T_j$.

В главе 6 приводится сравнительный анализ эффективности **Гибридного алгоритма** и алгоритма “Муравьиные колонии”.

В главах 7, 8 и 9 рассматриваются неклассические одноприборные задачи с обратными критериями оптимизации, одним невозобновимым ресурсом или обобщенной функцией запаздывания. В том числе, здесь перечислены результаты по задачам с целевой функцией – суммарное запаздывание.

В главе 10 представлена новая методика поиска “сложных” примеров.

В главе 11 рассмотрена качественно новая схема нахождения приближённых решений на основе интерполяционных полиномов Лагранжа.

Глава 2.
Свойства задачи $1||\sum T_j$

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

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

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

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

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

$$d_1 \le d_2 \le \cdots \le d_n,$$

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

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

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

(а) $t' + \sum\limits_{j=1}^{k} p_j < d_{k+1}$ (**правило исключения 1** [8, 11]) и

(б) $d_j + p_j \le t' + \sum\limits_{j=1}^{k} p_j$, 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 π^* обслуживания требований множества N, при котором $(j \to j^*)_{\pi^*}$ для всех требований $j \in \{1, 2, \ldots, k\} \setminus \{j^*\}$ и $(j^* \to j)_{\pi^*}$ для всех требований $j \in \{k+1, \ldots, n\}$ для некоторого $k \in L(N, t_0)$.* $\square$

Через $(i \to j)_\pi$ обозначают, что требование i обслуживается раньше требования j при расписании π.

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

Процедура ProcL (N, t)

0. Дан пример $\{N, t\}$ с множеством требований $N = \{j_1, j_2, \ldots, j_n\}$ и моментом начала обслуживания t, $d_{j_1} \le d_{j_2} \le \cdots \le d_{j_n}$;

1. IF $N = \emptyset$ **THEN** π^*:= 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
$N' := \{j_1, \ldots, j_k\} \setminus \{j^*\}$, $t' := t$,
$N'' := \{j_{k+1}, \ldots, j_n\}$, $t'' := t + \sum\limits_{i:=1}^{k} p_{j_i}$;

5. π^*:=$\arg \min\limits_{k \in L(N,t)} \{F(\pi_k, t)\}$;

6. RETURN π^*.

Алгоритм A
π^*:=**ProcL**(N, t_0).

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

Лемма 2 Правило исключения 4 [7, 13] *Если* $F(\pi^k) > F(\pi^{k+1})$ *или* $F(\pi^k) \ge F(\pi^i)$ *для требования* $j^* \le i < k$, *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* π^*, *при котором*

1. i *предшествует* j, $(i \to j)_{\pi^*}$, *если* $d_i \le \max(E_j, d_j)$ *и* $p_i \le p_j$;

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

Лемма 4 [6] *Пусть* π – *некоторое оптимальное расписание и*

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

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

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].

Глава 3.
Случай B задачи $1||\sum T_j$

Исследуется частный случай задачи, когда параметры требований удовлетворяют условиям:

$$\begin{cases} p_1 \geq p_2 \geq \cdots \geq p_n, \\ d_1 \leq d_2 \leq \cdots \leq d_n. \end{cases} \tag{1}$$

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

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

Процедура разбиения множества требований $N = \{j_1, \ldots, j_n\}, d_{j_1} \leq \cdots \leq d_{j_n}$, **на подмножества**

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

1. FOR $i = 2, 3, \ldots, n$ **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 $M_1, M_2, \ldots, M_k$, где $M_i = \{\alpha_i, \alpha_i+1, \ldots, \beta_i\}$, $\alpha_1 = j_1, \beta_k = j_n$. При этом, $M_1 \bigcup M_2 \bigcup \cdots \bigcup M_k = N$ и $M_i \bigcap M_j = \emptyset$, 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 B-1

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

$$\begin{cases} p_1 \geq p_2 \geq \cdots \geq p_n, \\ d_1 \leq d_2 \leq \cdots \leq d_n, \\ d_n - d_1 \leq p_n, \end{cases} \tag{2}$$

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

Далее в работе будем использовать следующие обозначения. Пусть

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

при этом, с учетом (2), для любого $t \in \mathbb{R}$ выполняется $d_1(t) \le d_2(t) \le \cdots \le d_n(t)$. Пусть $\pi_l(t)$ и $F_l(t)$ оптимальное расписание и соответствующее ему значение суммарного запаздывания для примера с подмножеством требований $N_l = \{l, \ldots, n\}$, моментом начала обслуживания равным 0, и директивными сроками окончания обслуживания $d_j(t)$, $j = l, \ldots, n$, $l = n, \ldots, 1$. Напомнить, что $F(\pi, t_0)$ – суммарное запаздывание требований множества N при расписании π, 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 $t \le t_0 + p_n$, тогда для примера $\langle \{p_j, d_j(t)\}_{j\in N}, 0\rangle$ будем иметь

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

В этом случае при любом расписании, построенном с момента времени 0, все требования множества N будут запаздывать, поэтому оптимальным будет SPT-расписание $(n, n-1, \ldots, 1)$. Если $t \ge t_0 + \sum\limits_{j=1}^{n} p_j$, тогда для примера $\langle \{p_j, d_j(t)\}_{j\in N}, 0\rangle$, учитывая $p_n \ge d_n - d_1$, будем иметь $t + p_n \ge t_0 + \sum\limits_{j=1}^{n} p_j + d_n - d_1$. Отсюда следует

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

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

Таким образом, при изменении t от $t_0 + p_n$ до момента времени $t_0 + \sum\limits_{j=1}^{n} p_j$ оптимальное расписание для примера $\langle \{p_j, d_j(t)\}_{j\in N}, 0\rangle$ изменяется от $(n, n-1, \ldots, 1)$ до $(1, 2, \ldots, n)$. 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 B-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** $l = n-1, n-2, \ldots, 1$:

$\pi^1 := (l, \pi_{l+1}(t - p_l))$, $\pi^2 := (\pi_{l+1}(t), l)$;
$F(\pi^1) := \max\{0, p_l - d_l(t)\} + F_{l+1}(t - p_l)$;
$F(\pi^2) := F_{l+1}(t) + \max\{0, \sum\limits_{j=l}^{n} p_j - d_l(t)\}$;
$F_l(t) := \min\{F(\pi^1), F(\pi^2)\}$;
$\pi_l(t) := \arg\min\{F(\pi^1), F(\pi^2)\}$.

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 B-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 доказывается, что такой способ построения позволяет получить оптимальное расписание для случая (2).

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

Теорема 2 *Расписание $\pi_1(d_n)$ является оптимальным для случая* (2). **Алгоритм B-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 $\{\pi_l\} = N_l = \{l, \ldots, n\}$, $\{\pi_1^*\} \bigcup \{\pi_2^*\} = \{1, \ldots, l\}$.

Предположим, что для всех оптимальных расписаний π существует такое $l \in N$, что $\pi = (\pi_1, j, \pi_2)$, где требование $j \notin N_l, \{\pi_1\} \bigcap N_l \neq \emptyset, \{\pi_2\} \bigcap N_l \neq \emptyset$. Выберем два требования α и β так, что $\alpha \in \{\pi_1\}$, $\beta \in \{\pi_2\}$, $\alpha, \beta \in N_l$. $\pi_1 = (\pi_{11}, \alpha, \pi_{12})$ и $\pi_2 = (\pi_{21}, \beta, \pi_{22})$, $\{\pi_{22}\} \bigcap N_l = \emptyset$, $\{\pi_{11}\} \bigcap N_l = \emptyset$, тогда расписание $\pi = (\pi_{11}, \alpha, \pi_{12}, j, \pi_{21}, \beta, \pi_{22})$. Из (2) мы имеем $p_1 \geq \cdots \geq p_n$ и $d_1 \leq \cdots \leq d_n$, 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 π построим расписание $\pi' = (\pi_{11}, \pi_{12}, j, \alpha, \pi_2)$. При расписаниях π и π' 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 α не запаздывают при обоих расписаниях, поэтому $F(\pi') \leq F(\pi)$. После такого сдвига требования α 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 $A = c_j(\pi) - p_j$, $B = \sum\limits_{j \in \{\pi_{21}\}} p_j$. Построим расписание $\pi' = (\pi_1, \beta, \pi_{21}, j, \pi_{22})$. При расписаниях π и π' выполняется.

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

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

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 π и π' 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 $\{\pi_1\} \bigcap N_l \neq \emptyset$, $\{\pi_2\} \bigcap N_l \neq \emptyset$, для некоторого l, $l = 1, \ldots, n$, можно построить соответствующее расписание π', 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 π может быть приведено к расписанию $\pi' = (\pi_1, \pi_l, \pi_2)$, где $\{\pi_l\} = N_l$ и, кроме того, $\{\pi_1\} \bigcup \{\pi_2\} = \{1, \ldots, l\}$, причем $F(\pi') \leq F(\pi)$. Таким образом, существует оптимальное расписание π^*, при котором

для всех $l = n, n-1, \ldots, 1$ и для всех $j \in N \setminus N_l$ выполняется либо $(j \to i)_{\pi^*}$, либо $(i \to j)_{\pi^*}$ для всех $i \in N_l$.

Покажем, что **алгоритм B-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 B-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 $(l \to j)_{\pi_l(t)}$, или верно $(j \to l)_{\pi_l(t)}$ 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 трудоёмкость **алгоритма B-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 $t \le t_0 + p_n$ и $t \ge t_0 + \sum\limits_{j=1}^{n} p_j$ 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 $\left[t_0 + p_n, t_0 + \sum\limits_{j=1}^{n} p_j\right]$. 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, трудоёмкость всего алгоритма составляет $O(n \sum p_j)$ операций. Теорема доказана. □

При реализации **алгоритма B-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 B-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 BF

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

Лемма 5 *Пусть для случая* (2) *известно, что при любом из* $n!$ *расписаний запаздывает ровно* m *требований. Существует оптимальное расписание, при котором требование* $j^* = \arg\max\limits_{j \in N}\{mp_j + d_j\}$ *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 $mp_{j^*} + d_{j^*} \ge mp_j + d_j$, то выполняется $F(\pi) \ge F(\pi')$. Следовательно существует оптимальное расписание π', при котором требование j^* запаздывает. □

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

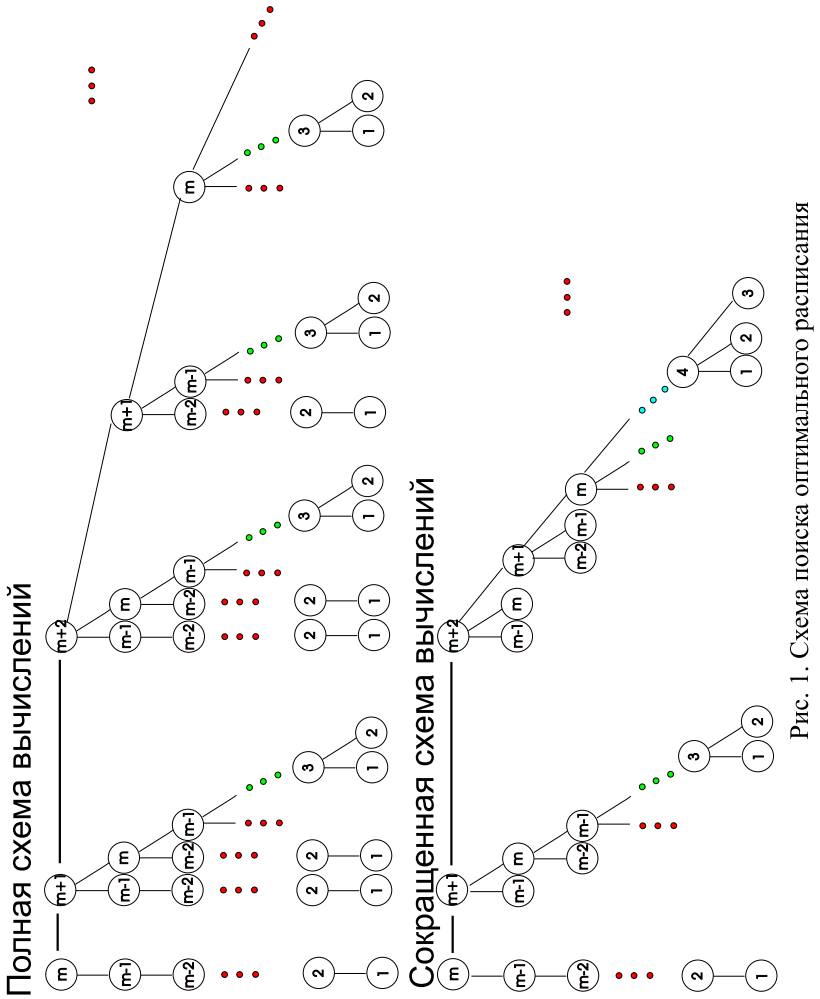


Рис. 1. Схема поиска оптимального расписания

Истинность леммы устанавливается леммой 10.

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

Обозначим через R множество частичных расписаний, WR – приоритет текущей "оптимальной" цепочки, TR – запаздывание требований текущей "оптимальной" цепочки, PR – продолжительность обслуживания цепочки. На предварительном шаге алгоритма задаём вектор $R(k)$ пустым.

Алгоритм BF.
$TR(k) := 0,\ WR(k) := -\infty,\ PR(k) := 0,\ R(k) := (),\ k = 1, \ldots, m+1.$
FOR $l = n - m + 1, n - m + 2, \ldots, n$:

FOR $k = m, m-1, \ldots, 1$:

$W := ((k-1)(PR(k-1) + p_{l-k}) -$
$(\sum p_j - PR(k-1) - d_{l-k}) - TR(k-1));$
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 $m, m-1$, $m-2, \ldots, 1$. То есть первое запаздывающее требование m, последнее запаздывающее требование 1. На каждом уровне $m, \ldots, 1$ 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 трудоёмкость **алгоритма BF** не превышает $O(n^2)$ операций.

3.3. Алгоритм B-1 модифицированный

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

Алгоритм B-1 модифицированный.
$F(t) := 0$, $\pi(t) := \emptyset$.
FOR $l = n, n-1, n-2, \ldots, 1$:

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

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

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

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

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

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 B-1 модифицированный** *может быть построено оптимальное расписание.*

Доказательство. При работе **алгоритма B-1** изменение частичного оптимального расписания происходит не в каждой точке t. В **алгоритме B-1 модифицированный** точки, где частичное оптимальное расписание не изменяется и не изменяется количество запаздывающих требований, не рассматриваются. То есть **алгоритм B-1 модифицированный** не проделывает "холостые" ходы. Поэтому алгоритмы эквивалентны с точки зрения процедуры нахождения решения. □

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

3.4. Случай B-1 general и алгоритм его решения

Рассматривается следующий случай:

$$\begin{cases} d_1 \le d_2 \le \cdots \le d_n, \\ d_n - d_1 \le p_{\min}, \end{cases} \tag{4}$$

где $p_{\min} = \min\limits_{j \in N} p_j$.

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

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

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

т.е. требования $j_1, j_2, \ldots, j_{k-1}$ при расписании π также не запаздывают. □

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

Доказательство. Пусть требование j_k запаздывает при расписании π, 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 $j_{k+2}, \ldots, j_n$. □

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* (π_{EDD}, π_{SPT}), π_{EDD} *и* π_{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 запаздывающие 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 π^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 $\pi' = (j_1, \ldots, j_{m-1}, j_m)$ из требований, которые обслуживаются при частичном расписании π^1.

Допустим, при расписании $\pi^{new} = (\pi', l, \pi^2)$, l – не первое запаздывающее требование. Покажем, что из π' 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 $j_1, j_2, \ldots, j_{m-2}$.

Учитывая, что $c(\pi^1)$ (момент завершения обслуживания частичного расписания π^1) неизменно, необходимо будет выбрать запаздывающим то требованием $i \in \{j_1, \ldots, j_m\}$, при котором $c(\pi^1) - d_i$ минимально, т.е. с наибольшим директивным сроком. Тогда частичное расписание π_{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 π_{EDD} будут не запаздывающими при расписании (π_{EDD}, l, π^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 π построим оптимальное расписание

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

где j_k – первое запаздывающее требование. Если $p_{j_k} \leq p_{j_{k+1}}$, то расписание имеет вид (π_{EDD}, π_{SPT}) и лемма формально верна. Если $d_{j_k} \geq d_{j_{k-1}}$, то расписание имеет вид (π_{EDD}, π_{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 $\pi^{**} = (j_1, \ldots, j_{k-1}, j_{k+1}, j_k, \ldots, j_n)$ 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 π^* не оптимально. Следовательно верно $c_{j_k}(\pi^*) \leq d_{j_{k+1}}$.

Рассмотрим расписание

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

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

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 π^{***} вида (π_{EDD}, π_{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 (π_{EDD}, π_{SPT}). Лемма доказана. □

Лемма 11 *Для случая* (4) *существует оптимальное расписание вида* $(\pi_{LPT}, l, \pi_{SPT})$, π_{LPT} *и* π_{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 $\{\pi_{EDD}\} = \{\pi_{LPT}\} \bigcup \{l\}$. Причем $F(\pi) = F(\pi')$. □

Алгоритм B-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** $l = n-1, n-2, \ldots, 1$:

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

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 B-1 general *на шаге* l *для множества требований* $N_l = \{l, l+1, \ldots, n\}$ *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 $N_{l+1} = \{l+1, \ldots, n\}$ построено оптимальное расписание $\pi_{l+1}(t)$.

Покажем, что на шаге l алгоритма для множества требований $N_l = \{l, l+1, \ldots, n\}$ 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 в точке t для множества требований $N_l = \{l, l+1, \ldots, n\}$ существует оптимальное частичное расписание π', при котором требование l не запаздывает, тогда частичное расписание обслуживания требований множества N_l вида $\pi^1 = (l, \pi_{l+1}(t - p_l))$ будет оптимальным.

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

   Рассмотрим расписание

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

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

   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 $(j_1, \ldots, j_m, j_{m+1}, \ldots, j_{n-l+1})$ 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 $F(\pi^2) \le F(\pi')$ или $F(\pi^1) \le F(\pi')$.

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

   Рассмотрим случай $T_l(\pi^2) > 0$.

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

   а) Если при расписании $(\pi'_\alpha, \pi'_\beta)$ требование j'_1 запаздывает. Предположим для расписаний π^2, π' 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 π_β 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 (π_α, π_β) – оптимальное частичное расписание на шаге $l+1$ в точке t.

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

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

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

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

Пусть $j_k \in \pi_\beta \bigcap \pi'_\alpha$ 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 π'' 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 π^1 или π^2 является оптимальным. □

Лемма 12 *Трудоёмкость* **алгоритма B-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, трудоёмкость всего алгоритма составляет $O(n^2 \sum p_j)$ операций. □

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

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

$$\begin{cases} d_1 \leq d_2 \leq \cdots \leq d_n, \\ d_n - d_1 \leq 1. \end{cases} \tag{5}$$

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

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

0. $S := t_0 + \sum\limits_{j=1}^{n} p_j$, $N' := N$, $\pi^* := \emptyset$, $\pi_D := (1, 2, \ldots, n)$;

1. WHILE $\{j \in N' : S - p_j \leq z\} = \emptyset$ **AND** $N' \neq \emptyset$ **DO:**

$\pi^* := (j^*, \pi^*)$, где $j^* := \arg\max\limits_{j \in N'} \left\{ d_j : p_j = \max\limits_{i \in N'} p_i \right\}$;

$S := S - p^*_j$, $N' := N' \setminus \{j^*\}$, $\pi_D := \pi_D \setminus \{j^*\}$;

2. IF $\mid N' \mid = 1$, 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 $\pi := \arg\min\limits_{i,j} F(\pi_{ij})$.

Оптимальность расписания π^*, построенного **алгоритмом C-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 $S := t_0 + \sum\limits_{j\in N'} p_j$. 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 $j^* = \arg\max\limits_{j\in N'} \left\{ d_j : p_j = \max\limits_{i\in N'} p_i \right\}$ 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 $d_i - d_j \le 1$ для всех $i, j \in N'$.

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

(а) $F(\pi') - F(\pi) = 0$, если $c_{j^*}(\pi) \ge d_j$;

(б) $F(\pi') - F(\pi) \le d_j - d_{j^*} \le 0$, 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 $c_j(\pi') \ge z + 1$. Следовательно, $F(\pi') - F(\pi) \le p_j - p_{j^*} < 0$.

2) Пусть $\{(\pi_2, \alpha)\} \ne \emptyset$. В этом случае имеем:
$T_j(\pi) = S - d_j$, $\quad T_j(\pi') = \max\{0, c_{j^*}(\pi) - p_{j^*} + p_j - d_j\}$,
$T_\alpha(\pi) = S - p_j - d_\alpha$, $\quad T_\alpha(\pi') = S - p_{j^*} - d_\alpha$,
$T_{j^*}(\pi) = \max\{0, c_{j^*}(\pi) - d_{j^*}\}$, $\quad T_{j^*}(\pi') = S - d_{j^*}$.
Рассмотрим следующие подслучаи:

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

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

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

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

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

Лемма 14 *Пусть параметры требований множества $N' \subseteq N$ удовлетворяют условиям* (5) *и $S := t_0 + \sum\limits_{j\in N'} p_j$, $\bar{N} := \{i \in N' : S - p_i \le z\}$. Если $\bar{N} \ne \emptyset$, тогда существует оптимальное расписание $\pi^* \in \Pi^*(N', t_0)$ такое, что выполняется $\pi^* = (\tilde{\pi}, \mu)$, где $c_\mu(\pi^*) - p_\mu \le z + 1$ 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 ν – некоторое требование из множества $\bar{N}$. Построим расписание $\pi' = (\pi_1, j, \pi_2, \alpha, \nu)$. Так как $j \notin \bar{N}$ и $S - p_\nu \le z$, то выполняется $p_\nu > p_j + 1$. Поскольку продолжительности обслуживания требований целочислены, то $p_\nu \ge p_j + 2$.

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

$T_\nu(\pi) = S - p_j - d_\nu$, $T_\nu(\pi') = S - d_\nu$,
$T_j(\pi) = S - d_j$, $T_j(\pi') = \max\{0, S - p_\nu - d_j\}$.
Из условия $p_\nu > p_j$ следует

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

2) Рассмотрим случай $\{(\pi_2, \alpha)\} \neq \emptyset$. В этом случае получаем
$T_\nu(\pi) = \max\{0, c_\nu(\pi) - d_\nu\}$, $T_\nu(\pi') = S - d_\nu$,
$T_j(\pi) = S - d_j$, $T_j(\pi') = 0$,
$T_\alpha(\pi) = S - p_j - d_\alpha$, $T_\alpha(\pi') = \max\{0, S - p_\nu - d_\alpha\}$.
Необходимо рассмотреть следующие подслучаи.

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

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

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

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

Следовательно, если $\bar{N} \neq \emptyset$, тогда $F(\pi') - F(\pi) \le 0$, 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 π', при котором требование ν обслуживается последним по порядку. Так как требование $\nu \in \bar{N}$, то будем иметь $S - p_\nu \le z < z + 1$. Лемма доказана. □

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

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 π и π' запаздывает не более одного требования (j и j_m, соответственно). Это следует из того, что $p_j \in \mathbb{Z}^+$, $\mid d_j - z \mid \le 1$ и $S \le z$.

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

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. □

Теорема 4 *Расписание π^*, 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 $\{i \in N' : S - p_i \le z\} = \emptyset$, и расписание π^* строится в соответствии с леммой 13.

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

Вычислим трудоёмкость 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 π_{ij} выбирается наилучшее. При каждом расписании π_{ij} запаздывает не более трех последних по порядку требований. Поэтому, чтобы вычислить значение величины $F(\pi_{ij})$ для каждого расписания π_{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. □

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

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

$$\begin{cases} d_1 \le d_2 \le \cdots \le d_n, \\ p_1 \ge p_2 \ge \cdots \ge p_n, \\ d_{\beta_1} - d_{\alpha_1} \le p_{\beta_1}, & \alpha_1 = 1, \\ d_{\beta_2} - d_{\alpha_2} \le p_{\beta_2}, & \alpha_2 = \beta_1 + 1, \\ \dots \\ d_{\beta_k} - d_{\alpha_k} \le p_{\beta_k}, & \beta_k = n. \end{cases} \tag{6}$$

Предлагается псевдо-полиномиальный алгоритм решения примеров задачи в случае (6). В данном параграфе будем использовать обозначения: $\Sigma_i(a) = \sum\limits_{j=1}^{i} a_j$ 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* $(l \to j)_{\pi^*}$ *для всех* $j \in \{l+1, l+2, \dots, n\}$*;*
- *или* $((l+1) \to l)_{\pi^*}$*.*

Доказательство. Очевидно, что для требований $n, n-1$ будем иметь или $(n \to (n-1))_{\pi^*}$, или $((n-1) \to n)_{\pi^*}$. Поэтому, рассмотрим случай $l \le n-2$. 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 $p_l \ge p_{l+1} \ge p_j$ и $d_l \le d_{l+1} \le d_j$.

Рассмотрим следующие случаи.

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

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

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

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

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

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

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

- *или* $(l \to j)_{\pi^*}$ *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_ν, 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 $(i \to l)_\pi$ и $(l \to j)_\pi$, т.е. $\pi = (\pi_1, i, \pi_2, l, \pi_3, j, \pi_4)$.

В силу условий (6) выполняется $p_i \le p_l$, $d_i \ge d_l$, а также $p_j \le p_l$, $d_j \ge d_l$ и $\mid d_i - d_j \mid \le p_{\beta_\nu}$. 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 $\mid M_\nu \setminus \{\alpha_m, \dots, l\} \mid \le 1$ (например, подмножество M_ν состоит из одного требования), то два требования i и j мы, очевидно, выбрать не сможем и лемма формально верна.

Рассмотрим следующие случаи.

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

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

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

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

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 π' или π''. Проведя подобные рассуждения последовательно для всех $l = n, n-1, \dots, 1$, мы придем к истинности леммы. □

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 $d'_j = \min \left\{ \max\{t_0 + p_j, d_j\}, t_0 + \sum\limits_{i=1}^{n} p_i \right\}$. Действительно, если $d_j > t_0 + \sum\limits_{i=1}^{n} p_i$, то $d'_j = t_0 + \sum\limits_{i=1}^{n} p_i$ 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 $F^*(x_1) - F^*(x_2) = \sum\limits_{j=1}^{n} t_0 + p_j - d_j = \text{const}$.

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

Если выполняется $t \le t_0 + p_n$, 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 $t \ge t_0 + 2\sum\limits_{j=1}^{n} p_j$, то учитывая неравенство $\sum\limits_{j=1}^{n} p_j \ge d_n - d_1$, получаем следующее очевидное неравенство $t + \sum\limits_{j=1}^{n} p_j \ge t_0 + 2\sum\limits_{j=1}^{n} p_j + d_n - d_1$. Отсюда следует

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

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 π, $\{\pi\} = N_l$, $N_l = \{l, l+1, \ldots, n\}$, $l = n, n-1, \ldots, 1$. Определим процедуру $G(\pi)$, которая производит разбиение расписания $\pi = (j_1, \ldots, j_m)$ на совокупность частичных расписаний $(\pi_1, \pi_2, \ldots, \pi_g)$, $g \le k$. При этом $\pi_i =$ **SubSequence** $(\pi \setminus \{\pi_{i+1}\} \bigcup \cdots \bigcup \{\pi_g\})$, $i = 1, \ldots, g$, где

$$\textbf{SubSequence}\ (\pi) := (j_\alpha, \ldots, j_m), \alpha = \arg \min_{1\le i\le m} \{\gamma(j_i) \ge \gamma(j_m)\}.$$

Если для некоторого $\pi_i = (j_\mu, \ldots, j_\nu)$ выполняется неравенство $\gamma(\pi_i) < \gamma(j_\nu)$, то будем полагать $G(\pi) = \emptyset$. Для SPT расписания $\pi = (n, n-1, \ldots, 1)$ имеем $G(\pi) = (\pi)$, и для EDD расписания $\pi = (1, 2, \ldots, n)$ имеем $G(\pi) = (\pi_1, \ldots, \pi_k)$, где $\{\pi_i\} = M_i$, $i = 1, \ldots, k$. Заметим, что если для некоторого расписания π выполняется $G(\pi) = \emptyset$, то существует расписание π', при котором $G(\pi') \ne \emptyset$ и верно $F(\pi') \le F(\pi)$. Справедливость этого следует из лемм 16 и 17.

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

Приведем алгоритм решения примеров в случае (6). Через ν_i, f_i, P_i, π_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 B-k 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** $m = k, k-1, \ldots, 1$; $l = \beta_m, \beta_m - 1, \ldots, \alpha_m$, $l < n$:

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

$$\pi^i := \left(\pi_1 \ldots, \pi_{i-1}, l, \pi_{\alpha_{\nu_i}}\left(t - \sum_{i-1}(P) - p_l\right)\right),$$

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

Пусть $i^* := \arg \min\limits_{i=1,2,\ldots,g+1} \{F(\pi^i)\}$, тогда $\pi_l(t) := \pi^{i^*}$, $F_l(t) := F(\pi^{i^*})$,

$$G_l(t) := \left\{\left\langle m, \sum_{i^*-1}(f) + \max\left\{0, \sum_{i^*-1}(P) + p_l - d_l(t)\right\}, \right.\right.$$
$$\left.\sum_{i^*-1}(P) + p_l, (\pi_1, \ldots, \pi_{i^*-1}, l)\right\rangle,$$
$$\left. G_{\alpha_{\nu_{i^*}}}\left(t - \sum_{i^*-1}(P) - p_l\right)\right\}.$$

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 B-k** *строит расписание $\pi_1(d_n)$ за $O(k\,n\sum p_j)$ операций.*

Доказательство. Каждый шаг **алгоритма B-k** организован в соответствии с леммами 16 и 17. Пусть $G(\pi_{l+1}(t)) = (\pi_1, \ldots, \pi_g)$. При работе алгоритма требование l ставится на обслуживание на следующие позиции: перед всеми требованиями множества N_{l+1}, после π_1, после π_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 "внутрь" любого π_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 B-k** выполняется за $O(kn\sum p_j)$ операций. Теорема доказана. □

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 B-1** и **B-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

$$d_j - d_{j-1} > p_j, \ j = 2, 3, \ldots, n, \tag{7}$$

без ограничений на продолжительность обслуживания требований и их целочисленность.

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

Процедура ProcB-n (N, t)

0. Дано множество требований $N = \{1, 2, \ldots, n\}$, момент начала обслуживания t, $d_1 < d_2 < \cdots < d_n$;

1. $S_\alpha := t + p_1 + p_2 + \cdots + p_\alpha$, $\alpha = 1, 2, \ldots, n$;
$j^* = \arg\max\limits_{j\in N}\{d_j : p_j = \max\limits_{i\in N} p_i\}$;

2. $\alpha^* := \arg\min\limits_{\alpha=1,\ldots,n} \{d_j + p_j \leq S_\alpha < d_{\alpha+1},\ j \in \{j^* + 1, \ldots, \alpha\}\}$;

3. $N' := \{1, \ldots, \alpha^*\} \setminus \{j^*\}$, $\quad t' := t$;
$N'' := \{\alpha^* + 1, \ldots, n\}$, $\quad t'' := S_{\alpha^*}$;

4. Оптимальное расписание $\pi^* := (\pi', j^*, \pi'')$, где частичные расписания π', π'' находятся рекурсивно $\pi' :=$ **ProcB-n** (N', t'), $\pi'' :=$ **ProcB-n** (N'', t'').

Алгоритм B-n

$\pi^* :=$ **ProcB-n** (N, t_0).

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

Лемма 18 *Если параметры требований d_j, p_j, $j \in N'$, удовлетворяют условиям (7), то существует оптимальное расписание обслуживания требований π^*, при котором требование j^* обслуживается на позиции α_1.*

Доказательство. Рассмотрим две соседние позиции α_i и α_{i+1} из множества подходящих позиций $L(N', t')$ для требования j^*, где $j^* = \arg\max\limits_{j \in N'} \left\{ d_j : p_j = \max\limits_{i \in N'} p_i \right\}$. Ниже будем обозначать $\alpha = \alpha_i$ и $\beta = \alpha_{i+1}$.

Обозначим через π_α и π_β расписания, при которых требование j^* обслуживается на позициях α и β, соответственно. То есть, $\pi_\alpha = (\pi_1, j^*, \pi_2)$, где π_1 есть оптимальное расписание обслуживания требований множества $\{1, \ldots, \alpha\} \setminus \{j^*\}$ с момента времени t', и π_2 есть оптимальное расписание обслуживания требований множества $\{\alpha + 1, \ldots, n\}$ с момента времени $S_\alpha = t' + \sum\limits_{i=1}^{\alpha} p_i$. Аналогично $\pi_\beta = (\pi_1', j^*, \pi_2')$, где π_1' есть оптимальное расписание обслуживания требований множества $\{1, \ldots, \beta\} \setminus \{j^*\}$ с момента времени t' и π_2' есть оптимальное расписание обслуживания требований множества $\{\beta + 1, \ldots, n\}$ с момента времени $S_\beta = t' + \sum\limits_{i=1}^{\beta} p_i$.

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 $d_\beta - d_{j^*} \geq p_{\alpha+1} + p_{\alpha+2} + \cdots + p_\beta = S_\beta - S_\alpha$. Получаем противоречие и $\beta \notin L(N', t')$, множество $L(N', t')$ содержит только одну позицию α_1 для требования j^*.

Рассмотрим случай, когда $d_{j^*} < S_\alpha$. Покажем, что выполняется $\pi_1' = (\pi_1, \bar{\pi})$, где $\bar{\pi} = (\alpha + 1, \alpha + 2, \ldots, \beta)$. 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 $\bar{\pi}' = (\alpha + 2, \alpha + 3, \ldots, \beta, \alpha + 1)$. Так как π_α является расписанием с наименьшим значением суммарного запаздывания среди множества расписаний, при которых требование j^* обслуживается на позиции α, 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 π' и π_β, и величина $\Delta = -\sum\limits_{j=\alpha+1}^{\beta} p_j = S_\alpha - S_\beta$.

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 $j \in \{\alpha+2, \ldots, \beta\}$, 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 α и β произвольные соседние позиции для требования j^*, то выполняется

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

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. □

Теорема 6 Алгоритм B-n *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 B-n**, являющийся модификацией **алгоритма A** на случай перебора только первой позиции из множества $L(N', t')$, строит оптимальное расписание в случае (7).

Каждый вызов процедуры **ProcB-n**(N, t) фиксирует позицию одного требования в оптимальном расписании π^* и выполняется за не более, чем $O(n)$ операций. Следовательно, **алгоритм B-n** выполняется за $O(n^2)$ операций. Теорема доказана. □

Глава 4.
Канонические примеры задачи $1||\sum T_j$

В этом разделе будут описаны два NP-трудных случая задачи $1||\sum T_j$ – канонические примеры DL [5] и LG. NP-трудность канонических примеров доказана сведением NP-полной задачи Чётно-Нечётного Разбиения к задаче $1||\sum T_j$.

4.1. Задача Чётно-Нечётного Разбиения (ЧНР).

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

Обозначим $\delta_i = b_{2i-1} - b_{2i},\ i = 1, \ldots, n$, $\delta = \sum\limits_{i=1}^{n} \delta_i$.

Построим модифицированный пример **Чётно-Нечётного Разбиения**.

$$\begin{cases} a_{2n} = M + b, \\ a_{2i} = a_{2i+2} + b,\ i = n-1, \ldots, 1, \\ a_{2i-1} = a_{2i} + \delta_i,\ i = n, \ldots, 1, \end{cases} \tag{8}$$

где $b \gg n\delta,\ M \ge n^3 b$.
Очевидно выполняется $a_i > a_{i+1},\ \forall i = 1, 2, \ldots, 2n-1$, кроме того $\delta_i = b_{2i-1} - b_{2i} = a_{2i-1} - a_{2i}$, $i = 1, \ldots, n$.

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

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

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

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

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

Построим *канонический пример* [5] задачи $\mathbf{1}\ |\ |\ \sum \boldsymbol{T_j}$ для множества из $3n+1$ требований $N = \{V_1, V_2 \ldots, V_{2n}, W_1, W_2, \ldots, W_{n+1}\}$. Пусть $b = (4n+1)\frac{1}{2}\delta$. Зададим параметры требований следующим образом:

$p_{V_i} = a_i, \qquad i = 1, 2, \ldots, 2n,$
$p_{W_i} = b, \qquad i = 1, 2, \ldots, n+1,$

$$d_{V_i} = \begin{cases} (j-1)b + \frac{1}{2}\delta + (a_2 + a_4 + \cdots + a_{2i}), & \text{если } i = 2j-1, \\ d_{V_{2j-1}} + 2(n-j+1)(a_{2j-1} - a_{2j}), & \text{если } i = 2j, \\ & j = 1, 2, \ldots, n, \end{cases}$$

$$d_{W_i} = \begin{cases} ib + (a_2 + a_4 + \cdots + a_{2i}), & \text{если } i = 1, 2, \ldots, n, \\ d_{W_n} + \frac{1}{2}\delta + b, & \text{если } i = n+1. \end{cases}$$

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

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

Теорема 7 [5] *Для примеров канонической задачи задачи* $\mathbf{1} \mid\mid \sum \boldsymbol{T_j}$, *к которой сводится* **ЧНР**, *существует оптимальное расписание, которое является каноническим.* □

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 для множества, состоящего из $3(n-l)+1$ требований $N_l = \{V_{2l-1}, V_{2l}, W_l, \ldots, V_{2n-1}, V_{2n}, W_n, W_{n+1}\}$ и директивными сроками окончания обслуживания $d_j(t)$, $j = V_{2l-1}, V_{2l}, W_l, \ldots, V_{2n-1}, V_{2n}, W_n, W_{n+1}$, $l = n, \ldots, 1$.

Алгоритм B-1 канонический

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

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

$\pi^1 := (V_{2l-1}, W_l, \pi_{l+1}(t - a_{2l-1} - b), V_{2l})$,
$\pi^2 := (V_{2l}, W_l, \pi_{l+1}(t - a_{2l} - b), V_{2l-1})$;
$F(\pi^1) := \max\{0, a_{2l-1} - d_{V_{2l-1}}(t)\} +$
$\max\{0, a_{2l-1} + b - d_{W_l}(t)\} + F_{l+1}(t - a_{2l-1} - b) +$
$\max\left\{0, \sum_{j=l}^{n}(a_{2j-1} + a_{2j} + b) + b - d_{V_{2l}}(t)\right\}$;
$F(\pi^2) := \max\{0, a_{2l} - d_{V_{2l}}(t)\} +$
$\max\{0, a_{2l} + b - d_{W_l}(t)\} + F_{l+1}(t - a_{2l} - b) +$
$\max\left\{0, \sum_{j=l}^{n}(a_{2j-1} + a_{2j} + b) + b - d_{V_{2l-1}}(t)\right\}$;
$F_l(t) := \min\{F(\pi^1), F(\pi^2)\}$;
$\pi_l(t) := \arg\min\{F(\pi^1), F(\pi^2)\}$.

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 δ.

Теорема 8 Алгоритм B-1 канонический *строит оптимальное каноническое расписание за время $O(n\delta)$ для канонических примеров задачи* $\mathbf{1}\,||\,\sum \boldsymbol{T_j}$, *к которым сводятся примеры* **ЧНР**.□

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

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

Показано, что частный случай **B-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 $\mathbf{1}\,||\,\sum \boldsymbol{T_j}$.

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

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

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

$$\begin{cases} p_1 > p_2 > \cdots > p_{2n+1}, & (9.1)\\ d_1 < d_2 < \cdots < d_{2n+1}, & (9.2)\\ d_{2n+1} - d_1 < p_{2n+1}, & (9.3)\\ p_{2n+1} = M = n^3 b, & (9.4)\\ p_{2n} = p_{2n+1} + b = a_{2n}, & (9.5)\\ p_{2i} = p_{2i+2} + b = a_{2i},\ i = n-1, \ldots, 1, & (9.6)\\ p_{2i-1} = p_{2i} + \delta_i = a_{2i-1},\ i = n, \ldots, 1, & (9.7)\\ d_{2n+1} = \sum\limits_{i:=1}^{n} p_{2i} + p_{2n+1} + \frac{1}{2}\delta, & (9.8)\\ d_{2n} = d_{2n+1} - \delta, & (9.9)\\ d_{2i} = d_{2i+2} - (n-i)b + \delta,\ i = n-1, \ldots, 1, & (9.10)\\ d_{2i-1} = d_{2i} - (n-i)\delta_i - \varepsilon\delta_i,\ i = n, \ldots, 1, & (9.11) \end{cases} \tag{9}$$

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

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

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

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) задачи $1||\sum T_j$

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 $\sum\limits_{i\in N'} p_i > (n+2)p_{\min} = (n+2)n^3 b$, где $p_{\min} = \min\limits_{j\in N}\{p_j\} = p_{2n+1}$.

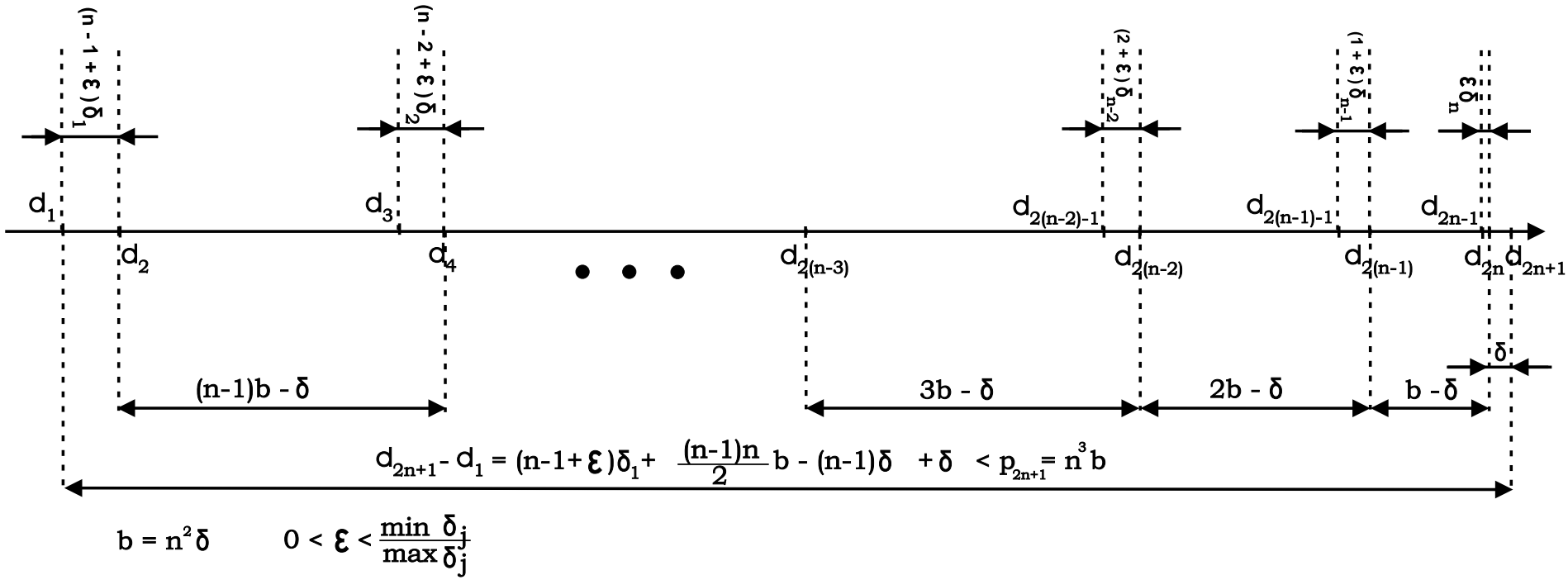


Рис. 2. Директивные сроки канонического LG-примера

Из (9.4)–(9.8) найдем

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

для которого будет выполняться

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

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 π 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 $N'' = \{V_1, V_2, \ldots, V_{2k-1}, V_{2k}\}$, будем иметь

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

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

   Из (9.8)–(9.11) будем иметь

$$d_{\min} = \min_{j \in N}\{d_j\} = d_1 = d_{2n+1} - \left(\sum_{i:=1}^{n-1}((n-i)b - \delta) + \delta + (n-1)\delta_1 - \varepsilon\delta_1\right) =$$

$$(n+1)n^3 b + (b + 2b + \cdots + nb) + \frac{1}{2}\delta -$$

$$\left(\sum_{i:=1}^{n-1}((n-i)b - \delta) + \delta + (n-1)\delta_1 + \varepsilon\delta_1\right) > P(N'');$$

   б) пусть $n = 2k+1$, тогда

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

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

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

$$d_{\min} = d_1 = d_{2n+1} - \left(\sum_{i:=1}^{n-1}((n-i)b - \delta) + \delta + (n-1)\delta_1 - \varepsilon\delta_1\right) = (n+1)n^3 b +$$

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

т.е. при любом расписании первые n требований будут не запаздывать. Таким образом, при любом расписании π количество незапаздывающих требований больше или равно n.

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

Лемма 21 *Если* (9)*, то для каждого расписания вида* $\pi = (\pi_1, \pi_2)$ *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 π_1. Так как n первых требований при π_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 $j = \arg\max\{d_i : i \in \{\pi_1\}\}$.

Рассмотрим частичное расписание π_2. Так как все n требований при π_2 запаздывают, то порядок SPT обслуживания требований множества $\{\pi_2\}$ будет оптимальным. □

Расписание вида

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

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

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* π_1 *и* π_2*, соответственно, то при расписании* π *некоторая пара требований* $\{V_{2i-1},\ V_{2i}\},\ i < n$*, не запаздывает или расписание* π *имеет вид*

$$(V_{1,1}, \ldots, V_{i,1}, \ldots, V_{n-1,1}, V_{2n-1}, V_{2n}, V_{2n+1}, V_{n-1,2} \ldots, V_{i,2}, \ldots, V_{1,2}), \tag{10}$$

т.е. пара требований $\{V_{2n-1}, V_{2n}\}$ *обслуживается до требования* V_{2n+1}*, одно требование из каждой пары* $\{V_{2i-1}, V_{2i}\}$*,* $i = 1, \ldots, n-1$*, 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 вида $\pi = (\pi_1, \pi_2)$, где $|\{\pi_1\}| = (n+1),\ |\{\pi_2\}| = n$. Возможны следующие случаи.

1. Если $\{\pi_2\} = \{V_{1,2}, \ldots, V_{n,2}\}$, т.е. при π_2 упорядочены n требований по одному требованию из всех n пар $\{V_{2i-1}, V_{2i}\},\ i = 1, \ldots, n$. Упорядочим требования из π_2 по правилу SPT, а требования из π_1 по правилу EDD. Тогда получим каноническое расписание π', причем $F(\pi') \leq F(\pi)$, согласно лемме 21.

2. Если $\{\pi_2\} \neq \{V_{1,2}, \ldots, V_{n,2}\}$, 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 $\{V_{2i-1}, V_{2i}\} \subset \{\pi_1\}$. □

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

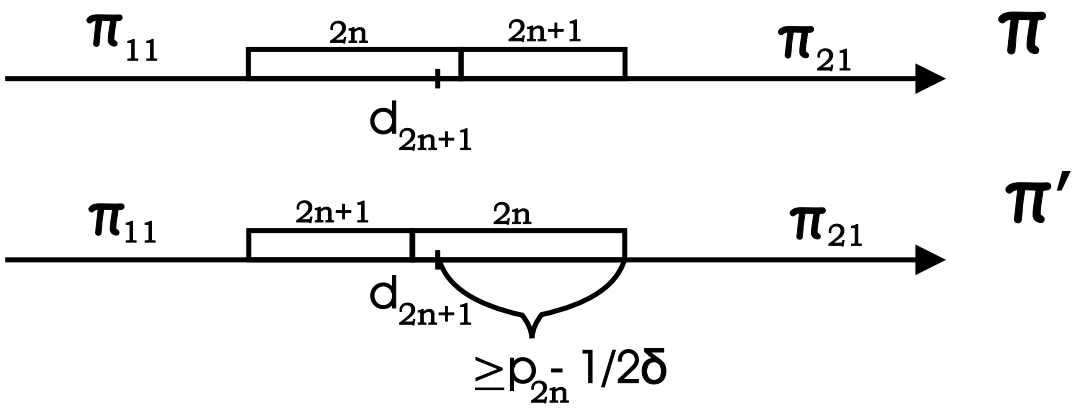


Рис. 3. Перестановка требований V_{2n} и V_{2n+1}

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

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

Будем иметь для требований подмножества $\{V_2, V_4, \ldots, V_{2i}, \ldots, V_{2n-2}, V_{2n-1}\}$

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

согласно (9.8). Далее, очевидно,

$$P(\{V_{1,1}, V_{2,1}, \ldots, V_{i,1}, \ldots, V_{n-1,1}, V_{2n-1}\}) + p_{2n} \geq$$

$$P(\{V_2, V_4, \ldots, V_{2i}, \ldots, V_{2n-2}, V_{2n-1}\}) + p_{2n},$$

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 π запаздывает.

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

а) Пусть при расписании π' 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 π' каноническое.

Из рис. 3 видно, что

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

б) Пусть при расписании π' требование V_{2n+1} запаздывает, тогда

$F(\pi) - F(\pi') = T_{2n}(\pi) + T_{2n+1}(\pi) - (T_{2n}(\pi') + T_{2n+1}(\pi')) = p_{2n} - p_{2n+1} = b > 0.$ □

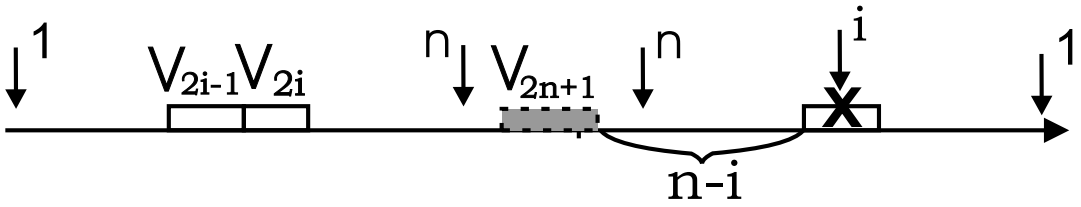


Рис. 4. Перестановка в неканоническом расписании

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 π 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 π' 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 $p_{2i} - p_X = ((j-i)b - \delta_j)$,

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

Следовательно,

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

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 π 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 π' 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 $p_X - p_{2i} = (i-j)b$,

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

Следовательно,

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

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

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

Следовательно,

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

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 π 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 π перед требованием V_{2i} будет запаздывать не меньше $(n-i)$ требований. Поэтому

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

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,

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

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 π – некоторое произвольное расписание. Согласно лемме 21 можно рассматривать только расписание следующего 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 π не каноническое.

Так как π не является каноническим расписанием, то в силу леммы 22 имеем при расписании π пару требований $\{V_{2i-1},\ V_{2i}\}$, которые не запаздывают, $i < n$, или расписание π 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 $\pi' = (V_{1,1}, \ldots, V_{i,1}, \ldots, V_{n-1,1}, V_{2n-1}, V_{2n+1}, V_{2n}, V_{n-1,2} \ldots, V_{i,2}, \ldots, V_{1,2})$, выполняется $F(\pi) > F(\pi')$. Переобозначим $\pi := \pi'$.

Далее опишем алгоритм, состоящий из двух циклов, преобразования исходного расписания π к каноническому виду.

Цикл 1. Пока среди не запаздывающих требований при очередном расписании π 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 π упорядочим по правилу **EDD**.

Требование V_{2n+1} занимает или позицию $(n+1)$ или позицию $n+2$ при расписании π. 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 некоторое 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 π_1 и π_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 π.

Примером рассматриваемой ситуации может служить расписание, при котором множества:

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

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

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

$$-(\delta - \delta_{i-1} - \delta_i - \delta_n) < P(\{V_{1,1}, V_{2,1}, \ldots, V_{i-2,1}, V_{i+1,1}, \ldots, V_{n-1,1}\}) -$$

$$P(\{V_{1,2}, V_{2,2}, \ldots, V_{i-2,2}, V_{i+1,2}, \ldots, V_{n-1,2}\}) < \delta - \delta_{i-1} - \delta_i - \delta_n$$

и выполняется

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

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

а) Пусть $q = 1$.

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

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

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

Поэтому

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

тогда

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

Известно, что $L + n^3 b = d_{2n+1}$, 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 π' следующего вида $\pi' = (\pi_1, V_{2n-1}, V_{2n+1}, V_{2n}, \pi_2)$.

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

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 π.

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

   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 π 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
   $F(\pi)-F(\pi')=(d_{2i}-d_{2(i-1)})-(n-i+1)(p_{2(i-1)}-p_{2i})=(n-i+1)b-\delta-(n-i+1)b=-\delta$,
   т.е. значение целевой функции увеличилось на δ.
   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 π', 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 π' 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 π 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 π за $O(n)$ операций можно преобразовать к каноническому расписанию π^*. Причем $F(\pi) > F(\pi^*)$. □

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

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

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

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

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

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

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

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

1. Если $V_{i,2} = V_{2i},\ i = 1, \ldots, n$, то $T_{V_{2n+1}}(\pi) = \frac{1}{2}\delta$,
   $\Phi_1 = \sum\limits_{i:=1}^{n-1}(n-i)p_{2i} - \frac{1}{2}\delta$.

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

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

Если решение модифицированного примера **ЧНР** "НЕТ", то не выполняется равенство $\sum\limits_{i:=1}^{n}\phi(i)\delta_i = \frac{1}{2}\sum\limits_{i:=1}^{n}\delta_i$. 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 $\sum\limits_{i:=1}^{n} p_{V_{i,1}} = \sum\limits_{i:=1}^{n} p_{V_{2i}} + \frac{1}{2}\delta = \sum\limits_{i:=1}^{n} p_{V_{i,2}}$, т.е. решение модифицированного примера **ЧНР** будет иметь ответ "ДА". $\square$

Глава 5.
Трудоёмкость известных алгоритмов

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

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 трудоёмкость (от количества требований n) для *канонических* случаев задачи $1||\sum T_j$.

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

Частный случай BF, для которого существует алгоритм трудоёмкости $O(n^2)$, анализируется в главе 3.

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 трудоёмкость.

Определение. *Канонические примеры, для которых выполняется*

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

будем называть примерами случая SD (short delta).

Для случая *SD* выполняется

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

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

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): $(V_1, V_2, W_1, \ldots, V_{2i-1}, V_{2i}, W_i, \ldots, W_n, W_{n+1})$.

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 только правил исключения 1–3, то 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.*

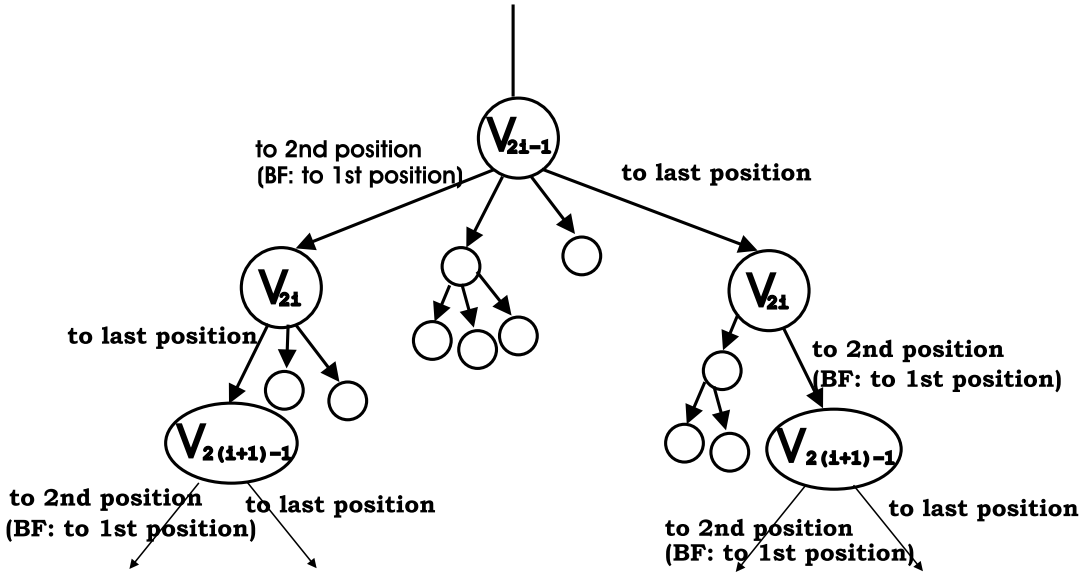


Рис. 5. Дерево поиска

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,

   $$N = \{V_1, V_2, W_1, \ldots, V_{2i-1}, V_{2i}, W_i, \ldots, W_n, W_{n+1}\}.$$

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

   a. 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

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

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

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

   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

$$0 + \sum_{j=1}^{3n+1} p_j = \sum_{j=1}^{2n} a_j + (n+1)b >$$

$$(n+1)b + \frac{1}{2}\delta + a_2 + a_4 + \cdots + a_{2n} + a_1.$$

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

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 $1, 2, \ldots, i-1$. 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:

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

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

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

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

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

"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 $t' + p_{W_{i-1}} + p_{V_{2i-1}} =$
$t' + b + a_{2i-1} = a_2 + a_4 + \cdots + a_{2i-2} + (i-2)b + \bar{\delta} + a_{2i-1} +$
$b > d_{W_{i-1}} + p_{W_{i-1}} = a_2 + a_4 + \cdots + a_{2i-2} + (i-2)b + b.$

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

Для случая SD выполняется: $t' + b + a_{2i-1} = a_2 + a_4 + \cdots + a_{2i-2} + (i-1)b + \bar{\delta} + a_{2i-1} <$ $(i-1)b + \frac{1}{2}\delta + (a_2 + a_4 + \cdots + a_{2i-2} + a_{2i}) + 2(n-i+1)(a_{2i-1} - a_{2i}) = d_{V_{2i}}$.

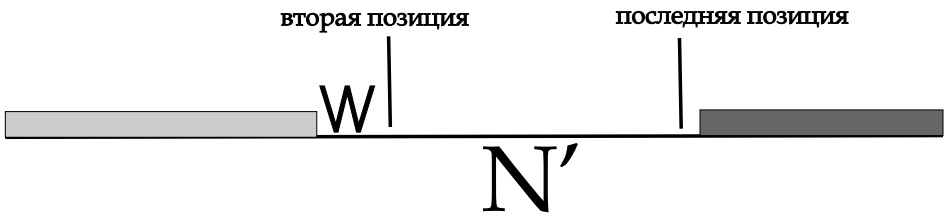


Рис. 6. Текущие “вторая” и “последняя” позиции

Покажем, что при постановке V_{2i-1} во “вторую” (после W_{i-1}) позицию для требования V_{2i} становится подходящей “последняя” позиция в списке $N'' = \{V_{2i}, W_i, \ldots, W_n, W_{n+1}\}$. Правило **b** также выполняется:
$t' + b + a_{2i-1} + \sum\limits_{j \in N''} p_j = a_2 + a_4 + \cdots + a_{2i-2} +$

$(i-1)b + \bar{\delta} + (n-i+2)b + \sum\limits_{j:=2i-1}^{2n} a_j > a_2 + a_4 + \cdots + a_{2i-2} + a_{2i} + a_{2i+2} + \cdots + a_{2n} + (n+1)b + a_{2i}$,
поэтому

$a_2 + a_4 + \cdots + a_{2i-2} + a_{2i} + a_{2i+2} + \cdots + a_{2n} + (n+1)b + a_{2i} = d_{W_{n+1}} + a_{2i} = \max\limits_{j \in N''} d_j + \max\limits_{j \in N''} p_j >$
$d_j + p_j, \forall j \in N''$.
Правило **a** для “последней” позиции, очевидно, выполняется, т.к. директивный срок “фиктивного требования” $d_{|N''|+1} = +\infty > t' + b + a_{2i-1} + \sum\limits_{j \in N''} p_j$.

б. Покажем, что текущая “последняя” позиция для требования V_{2i-1} является также подходящей.
Правило **a**, очевидно, выполняется, т.к. имеем $d_{|N'|+1} = +\infty > t' + \sum\limits_{j \in N'} p_j$.
Покажем выполнение правила **б**, т.к.
$t' + \sum\limits_{j \in N'} p_j = a_2 + a_4 + \cdots + a_{2i-2} + (i-1)b + \bar{\delta} +$
$(n-i+2)b + \sum\limits_{j:=2i}^{2n} a_j > a_2 + a_4 + \cdots + a_{2i-2} +$
$a_{2i} + a_{2i+2} + \cdots + a_{2n} + (n+1)b + a_{2i-1}$, поэтому

$a_2 + a_4 + \cdots + a_{2i-2} + a_{2i} + a_{2i+2} + \cdots + a_{2n} + (n+1)b +$
$a_{2i-1} = d_{W_{n+1}} + a_{2i-1} = \max_{j \in N'} d_j + \max_{j \in N'} p_j >$
$d_j + p_j, \forall j \in N'$, 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 $N'' = \{W_{i-1}, V_{2i}, W_i, \ldots, W_n, W_{n+1}\}$.
Правило **б** выполняется, т.к. $t' + b + a_{2i} = a_2 + a_4 + \cdots + a_{2i-2} + (i-2)b + \bar{\delta} + b + a_{2i} >$
$d_{W_{i-1}} + p_{W_{i-1}} = a_2 + a_4 + \cdots + a_{2i-2} + (i-2)b + b$.
Покажем, что выполняется и правило **a**:
$t' + b + a_{2i} = a_2 + a_4 + \cdots + a_{2i-2} + (i-1)b + \bar{\delta} + a_{2i} <$
$ib + (a_2 + a_4 + \cdots + a_{2i-2} + a_{2i}) = d_{W_i}$, т.к. верно $b \gg \delta > \bar{\delta}$. □

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 выполняется

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

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

   Пусть $\pi_1 = (W_{i-1}, V_{2i-1}, V_{2i}, W_i, \ldots, W_n, W_{n+1})$,
   $\pi_2 = (W_{i-1}, V_{2i}, V_{2i-1}, W_i, \ldots, W_n, W_{n+1})$.

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

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

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

   б. $t' + b + a_{2i} - d_{V_{2i}} \le 0$ и $t' + b + a_{2i-1} - d_{V_{2i-1}} > 0$.
   $t' + b + a_{2i} - d_{V_{2i}} \le 0 \Rightarrow a_2 + a_4 + \cdots + a_{2i-2} + a_{2i} + (i-1)b + \bar{\delta} \le (i-1)b + (a_2 + a_4 + \cdots + a_{2i-2} + a_{2i}) + \frac{1}{2}\delta + 2(n-i+1)(a_{2i-1} - a_{2i}) \Rightarrow \bar{\delta} \le \frac{1}{2}\delta + 2(n-i+1)(a_{2i-1} - a_{2i})$.

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

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

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

   Рассмотрим частичное расписание требований множества $N'\pi = (W_{i-1}, V_{2i}, W_i, \ldots, W_n, W_{n+1}, V_{2i-1})$, обслуживание которого начинается с момента времени t'. Очевидно, что $d_{V_{2i}} \le t' + p_{W_{i-1}} + p_{V_{2i}}$, $d_{W_i} \le t' + p_{W_{i-1}} + p_{V_{2i}} + p_{W_i}$, $d_{W_{i-1}} \le t' + p_{W_{i-1}}$. Все остальные требования из π 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 π и π' требования из π_2 и требование V_{2i-1} запаздывают.

При расписании π' суммарное запаздывание требований из π_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 $F(\pi) < F(j^*, i), \quad j^* \le i \le k$.

□

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

Доказательство. Пусть

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

Имеем множество требований

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

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':

$\pi_1 = (W_{i-1}, V_{2i}, W_i, V_{2(i+1)-1}, V_{2(i+1)}, \ldots, V_{2i-1}, X, \ldots, W_{n+1})$ и

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

Не трудно показать, что все требования из π_1 и π_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 $X \in N' \backslash V_{2i-1}$. □

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 $(W_{i-1} \to j)_\pi, \ \forall j \in \{V_{2i-1}, V_{2i}, W_i, \ldots, W_{n+1}\}$ при любом каноническом DL-расписании. Тогда $E_{V_{2i-1}} = (i-1)b + a_{2i-1}, \ E_{V_{2i}} = (i-1)b + a_{2i}, \ E_{W_i} = ib = d_{W_i}$. □

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

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

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

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

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

Величины b_i не влияют на ветвления основного дерева, влияют только $\delta_i, \ i = 1, 2, \ldots, n$. 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. Трудоёмкость 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 трудоёмкость не меньше $O(n2^{\frac{n-1}{3}-1})$ операций.*

Доказательство. В лемме 27 показано, что для очередного требования V_{2i-1}, $i = 1, 2, \ldots, n-1$ "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 трудоёмкость $O(n)$ операций.

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

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 чисел $k, l = 1, \ldots, n, k \neq l$, то сохранится, по крайней мере, "половина дерева" (ветвление имеет место для требований $i = 1, \ldots, n/2$). Таким образом экспоненциальная трудоёмкость алгоритма сохранится.

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

5.3. Трудоёмкость известных алгоритмов для случая (2)

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

$$\begin{cases} p_1 \geq p_2 \geq \cdots \geq p_n, \\ d_1 \leq d_2 \leq \cdots \leq d_n, \\ d_n - d_1 \leq p_n, \\ n = 2k, \\ \sum\limits_{i:=1}^{k} p_i < d_j < \sum\limits_{i:=k}^{n} p_i, \ j = 1, 2, \ldots, n, \\ p_1 - p_n \ll p_n, \\ \sum\limits_{i:=k+j+1}^{n} (p_{2j-1} - p_i) > d_{k+j} - d_{2j}, j = 1, \ldots, (k-1), \\ \sum\limits_{i:=k+j+1}^{n} (p_{2j} - p_i) > d_{k+j} - d_{2j}, j = 1, \ldots, (k-1). \end{cases} \tag{11}$$

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

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, $i = 1, \ldots, (k-1)$. 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 $1, 2, \ldots, i-1$, $i < k$.

Рассматривается подмножество из $n-2i+2$ требований $N' \subseteq N$, $N' = \{V_{2i-1}, V_{2i}, \ldots, V_n\}$.

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

Требование с максимальной продолжительностью обслуживания $j^* = V_{2i-1}$. 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**:

$t' + p_{2i-1} = p_2 + p_4 + \cdots + p_{2i-2} + \bar{\delta} + p_{2i-1} < d_{2i}$. 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. Рассмотрев 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. директивный срок фиктивного требования $d_{|N'|+1} = +\infty > t' + \sum_{j \in N'} p_j$.

Покажем выполнение правила **б**.

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

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

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 π'' требование V_{2i-1} запаздывает. Очевидно, что при расписаниях π' и π'' 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 π'' 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 π' 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 $\{\pi_{22}\} = \{V_{n-|\{\pi_{22}\}|+1} \ldots, V_n\} = \{V_{k+i+1} \ldots, V_n\}$. Тогда при расписании π'' стало запаздывать еще одно требование $V_{k+i} \in \{\pi_{21}\}$.

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

   Для случая (11) выполняется $d_{k+i} - d_{2i-1} < \sum\limits_{j:=k+i+1}^{n} (p_{2i-1} - p_j)$, то $F(\pi') - F(\pi'') < 0$.

   Тогда $F(V_{2i-1}, i) > F(V_{2i-1}, |N'|),\ 1 \le i < |N'|$. 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. □

Нетрудно показать, что $E_j = p_j,\ L_j = \sum\limits_{i:=1}^{n} p_i$. 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 трудоёмкость $O(n2^{n/2})$.*

Доказательство.

Аналогично доказательству теоремы 11. □

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

Глава 6.
Метаэвристический подход

Нами построен **гибридный алгоритм**, использующий идею известного метаэвристического **алгоритма "Муравьиные колонии" (ACO)** [14] и комбинаторные свойства правил исключения 1–4. В разд. 6.1. приводится **алгоритм ACO**. Далее описывается **гибридный алгоритм** и приводиться сравнительный анализ его эффективности и эффективности **алгоритма ACO**.

6.1. Алгоритм ACO 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 (ACO) [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 $i = 1, 2, \ldots, n$ выбирается требование j для подстановки на место i расписания используя "вероятности перехода".

В алгоритме используются параметры:

η_{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 $i = 1, \ldots, n$ 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: $\eta_{ij} = \frac{1}{p_j}$,

τ_{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 η_{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 η_{ij} рассчитываются один раз перед первой итерацией.

На каждом шаге вычисляется **матрица вероятностей перехода**:

$$\rho_{ij} = \begin{cases} \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum\limits_{h\in\Omega}[\tau_{ih}]^\alpha [\eta_{ih}]^\beta}, & j \in \Omega, \\ 0, & j \notin \Omega, \end{cases}$$

где Ω – множество неупорядоченных требований.

Правило, по которому на позицию i выбирается требование j, определяется следующим образом:

$$\begin{cases} j = \arg\max\limits_{h\in\Omega}[\tau_{ih}]^\alpha [\eta_{ih}]^\beta, & q < q_0, \\ j \text{ определяется случайным образом} \\ \text{согласно распределению вероятностей } \rho_{ij}, & q \geq q_0, \end{cases}$$

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" τ_{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 ACO дает хорошие результаты при интеграции в него локального поиска, к примеру, попарной перестановки требований. Локальный поиск запускается после каждой итерации.

В работе [14] приводиться 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 трудоёмкость алгоритма без локального поиска $O(mn^2)$. Для каждой позиции i (всего n позиций) выбиралось требование j за $O(n)$ операций.

Трудоёмкость локального поиска $O(n^3)$. Тогда трудоёмкость **алгоритма ACO** с локальным поиском составляет не меньше $O(mn^3)$ операций.

6.2. Гибридный алгоритм

На основе **алгоритма ACO** 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 подходящую позицию $k \in L(N,t)$.

После каждой итерации "глобальный след" τ_{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.

Модифицированная процедура ProcL (N,t)

0. Дан пример $\{N,t\}$: необходимо обслужить множество требований $N = \{j_1, j_2, \ldots, j_n\}$ с момента начала обслуживания t, $d_{j_1} \le d_{j_2} \le \cdots \le d_{j_n}$;

1. **IF** $N = \emptyset$ **THEN** π^*:= 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 $i \in L(N,t)$:

$$\rho_{ij*} = \frac{\tau_{ij*} 1/F(\pi^i)}{\sum\limits_{h \in L(N,t)} \tau_{hj*} 1/F(\pi^h)},$$

где $\pi^i = (j_1, \ldots, j_{m-1}, j_{m+1}, \ldots, j_i, j^*, j_{i+1}, \ldots, j_n)$, требование $j^* = j_m,\ m < i$;

5. Выберем позицию $k \in L(N,t)$ произвольным образом согласно распределению вероятностей ρ_{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**

π^* :=(**ProcL**(N',t'), j^*,**ProcL**(N'',t'')), где
$N' := \{j_1, \ldots, j_k\} \setminus \{j^*\}$, $t' := t$,
$N'' := \{j_{k+1}, \ldots, j_n\}$, $t'' := t + \sum\limits_{i:=1}^{k} p_{j_i}$.

После каждой итерации запускается процедура локального поиска – попарная перестановка требований.

Не трудно убедиться, что трудоёмкость алгоритма без локального поиска $O(mn^2)$. Процедура ProcL запускается n раз. В каждой процедуре выполняется порядка $O(n)$ действий.

Трудоёмкость локального поиска $O(n^3)$. Тогда трудоёмкость **гибридного алгоритма** с локальным поиском составляет не меньше $O(mn^3)$ операций. То есть теоретическая трудоёмкость исследуемых алгоритмов идентична.

6.3. Эффективность алгоритмов для тестовых примеров [9]

Вычислительные эксперименты, представленные в этом параграфе, проводились для примеров размерности $n = 4, \ldots, 70, 100$, 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 **ACO** гарантированно находят точные решения. π_{EDD} – расписание построенное по правилу EDD.

В алгоритмах использовались следующие значения параметров: $\alpha = 1$; $\beta = 2$; $\rho = 0,1$. Локальный поиск – попарная перестановка. Применяемая эвристика – MDD [14].

Для каждого примера запускался точный **алгоритм А**. Мы получали точное решение F_{opt}.

В **алгоритме ACO** муравьи генерировались до тех пор, пока не находилось расписание π, 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

№	Кол. прим.	не опт. ACO	не опт. Гиб.	rel. ACO	rel. Гиб.	Мур. ACO	Мур. Гиб.
1	2	3	4	5	6	7	8
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

Окончание табл. 1.

1	2	3	4	5	6	7	8
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
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
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 ACO** 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 ACO**, 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 ACO** 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**.

6.4. Эффективность алгоритмов для случая B-1

В этом параграфе представлены экспериментальные исследования алгоритмов для примеров случая B-1 2.

Эксперименты, аналогичные экспериментам из предыдущего параграфа, проводились для примеров размерности $n = 4, \ldots, 100$. Для каждого значения n рассматривалось по 1000 примеров случая B-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 B-1 модифицированный**, трудоёмкость которого не больше $O(n \sum p_j)$ для целочисленных примеров.

Были получены следующие результаты.

Таблица 2. Результаты экспериментов на примерах B-1

№	Кол. прим.	не опт. ACO	не опт. Гиб.	rel. ACO	rel. Гиб.	Мур. ACO	Мур. Гиб.
1	2	3	4	5	6	7	8
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

Продолжение табл. 2

1	2	3	4	5	6	7	8
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
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
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

Окончание табл. 2

1	2	3	4	5	6	7	8
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
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 ACO 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 A**, ниже эффективности **алгоритма ACO** на примерах B-1. Можно объяснить это тем, что примеры B-1 "наиболее сложные" для **алгоритма A**.

Заметим также, что при такой генерации примеров, трудоёмкость точного **алгоритма B-1 модифицированный** меньше $O(n^3)$, так как генерацией "не захватывается" NP-трудный подслучай.

6.5. Эффективность алгоритмов для канонических DL-примеров [5]

Мы генерировали примеры ЧНР для $n = 4, \ldots, 40$. Параметры δ_i распределены по равномерному закону на промежутке $[1, 50]$. Пример задачи ЧНР задается следующим образом:

$b_{2n} := 1;\ b_{2n-1} := b_{2n} + \delta_n;\ b_{2i} := b_{2i+1} + 1;\ b_{2i-1} := b_{2i} + \delta_i;\ i := 1, \ldots, n-1.$

За полиномиальное время мы преобразовывали пример ЧНР к каноническому DL-примеру (количество требований $3n + 1 = 13, 16, \ldots, 121$). Для канонического DL-примера запускался точный псевдополиномиальный **алгоритм B-1 канонический**, трудоёмкостью $O(n\delta)$, **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
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 работы алгоритмов примерно идентичны. Только для $3n + 1 = 25,\ 28$ количество неточно решенных примеров достигает 10%. При этом погрешность алгоритмов

меньше 0.01%. В среднем 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 $V_{2i-1}, V_{2i},\ i := n, \ldots, 1$ 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".

Таблица 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
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.

Глава 7.
Минимизация обобщенной функции запаздывания

В данном параграфе рассматривается обобщение двух классических задач теории расписаний, целевые функции которых связаны с запаздыванием требований, $1||\sum w_jU_j$ и $1||\sum T_j$ – минимизация взвешенного числа запаздывающих требований и минимизация суммарного запаздывания для одного прибора.

В задаче $1||\sum w_jU_j$ для каждого требования $j = 1, 2, \ldots, n$, задан вес $w_j > 0$. Если $C_j(\pi) > d_j$, тогда требование j запаздывает при расписании π, и в этом случае полагают $U_j = 1$. Если $C_j(\pi) \le d_j$, тогда требование j не запаздывает, и $U_j = 0$. Необходимо построить расписание π, при котором значение целевой функции $F(\pi) = \sum\limits_{j=1}^{n} w_jU_j(\pi)$ минимально.

В обобщенной задаче для каждого требования j, $j = 1, 2, \ldots, n$, заданы *квота запаздывания* $b_j \ge 0$, *коэффициент нормального запаздывания* $v_j \ge 0$ и *коэффициент ненормального запаздывания* $w_j \ge 0$.

Обобщённое запаздывание задается следующим образом:

$$GT_j(\pi) = \begin{cases} 0, & \text{если} \quad C_j(\pi) - d_j \le 0, \\ v_j \cdot (C_j(\pi) - d_j), & \text{если} \quad 0 < C_j(\pi) - d_j \le b_j, \\ w_j, & \text{если} \quad b_j < C_j(\pi) - d_j, \end{cases}$$

где $w_j \ge v_jb_j$ для каждого $j \in N$. Целевая функция задачи:

$$F(\pi) = \sum_{j=1}^{n} GT_j(\pi).$$

Обобщённую функцию запаздывания можно интерпретировать следующим образом. Если запаздывание требования j превышает параметр b_j, то значение штрафа уже не зависит от запаздывания, остается постоянным и равным w_j. Необходимо найти расписание, минимизирующее значение $F(\pi)$. Эту задачу будем обозначать как $1||\sum GT_j$.

Очевидно, что эта задача NP-трудна, так как ее частный случай $b_j = 0$ соответствует NP-трудной задаче $1||\sum w_jU_j$.

В данном параграфе мы рассмотрим частный случай задачи, при котором

$$b_j = p_j, \quad v_j = 1, \quad w_j = p_j, \tag{12}$$

т.е., $GT_j(\pi) = \min\{\max\{0, C_j(\pi) - d_j\}, p_j\}$ для всех $j \in N$.

Теорема 13 *Частный случай* (12) *задачи* $1||\sum GT_j$ *является* NP*-трудным.*

Доказательство. Сведем к данному частному случаю ЗАДАЧУ РАЗБИЕНИЯ. Дан пример ЗАДАЧИ РАЗБИЕНИЯ. Построим пример частного случая (12) с $n + 1$ требованиями. Продолжительности обслуживания $p_j = a_j, j = 1, 2, \ldots, n$, и $p_{n+1} = 1$, т.е. $\sum\limits_{j=1}^{n+1} p_j = 2A + 1$. Директивные сроки $d_j = A$, $j = 1, 2, \ldots, n$, и $d_{n+1} = A + 1$. При любом расписании π выполняется $F(\pi) \ge A = \sum\limits_{j=1}^{n+1} p_j - d_{n+1}$. Более того, для всех расписаний выполняется $F(\pi) \le A + 1$.

При оптимальном расписании $\pi^* = (\pi_1, n+1, \pi_2)$ (без простоев) равенство $C_{n+1}(\pi^*) = A+1$ выполняется тогда и только тогда, когда ответ в примере ЗАДАЧИ РАЗБИЕНИЯ – "ДА". Все требования из частичного расписания π_1 и требование $n + 1$ не запаздывают, а требования из π_2 запаздывают. Причем, $F(\pi^*) = A$. $\square$

Лемма 33 *Для частного случая* (12) *существует оптимальное расписание вида* $\pi = (G, H) = (EDD, LDD)$*, где для всех требований* $i \in G$ *выполняется* $0 \le GT_i(\pi) < p_i$*, а для требований* $j \in H$ *выполняется* $GT_i(\pi) = p_i$*. Требования из множества* G *обслуживаются в порядке EDD, а требования из множества* H *– в порядке LDD.*

Доказательство. 1) Допустим, что существует оптимальное расписание $\pi^* = (\pi_1, j, \pi_2)$. Если $GT_j(\pi) = p_j$, тогда расписание $\pi' = (\pi_1, \pi_2, j)$ также оптимальное. То есть существует оптимальное расписание вида $\pi = (G, H)$, где все требования $j \in H$ запаздывают и $GT_j(\pi) = p_j$. Для всех требований $i \in G$ выполняется $0 \le GT_i(\pi) < p_i$.
2) Рассмотрим оптимальное расписание вида $\pi = (G, H)$, где все требования $j \in H$ запаздывают и $GT_j(\pi) = p_j$, а для всех требований $i \in G$ выполняется $0 \le GT_i(\pi) < p_i$. Докажем, что требования $i \in G$ обслуживаются в порядке EDD.

Предположим, что существует оптимальное расписание вида $\pi = (\pi_1, \alpha, \beta, \pi_2)$, где требования $\alpha, \beta \in G$, и выполняется $d_\alpha > d_\beta$. А также выполняется $C_\alpha(\pi) - d_\alpha < p_\alpha$ и $C_\beta(\pi) - d_\beta < p_\beta$, тогда $C_\alpha(\pi) < d_\alpha$.

Рассмотрим расписание $\pi' = (\pi_1, \beta, \alpha, \pi_2)$. Определим $C = C_\beta(\pi) = C_\alpha(\pi')$. Тогда

$$F(\pi) - F(\pi') = (GT_\alpha(\pi) - GT_\alpha(\pi')) + (GT_\beta(\pi) - GT_\beta(\pi')) =$$

$$-\min\{p_\alpha, \max\{0, C - d_\alpha\}\} + \min\{p_\alpha, \max\{0, C - d_\beta\}\} \ge 0$$

и расписание π' также оптимально.
3) Докажем, что требования $j \in H$ обслуживаются в порядке LDD. Для всех требований $j \in H$ имеем $d_j \le \sum_{l=1}^{n} p_l - \sum_{k \in H} p_k$, иначе, если $d_j > \sum_{l=1}^{n} p_l - \sum_{k \in H} p_k$, тогда расписание $\pi' = (G, j, H \setminus \{j\})$ лучше и имеет место противоречие. Поэтому требования из H могут быть обслужены в любом порядке. □

Этот результат аналогичен теореме 38. Пользуясь этой леммой, можно построить точный алгоритм решения данного частного случая, аналогичный алгоритму 1. Трудоёмкость алгоритма будет равна $O(nd_{\max})$ операций, где $d_{\max}$ – максимальный директивный срок.

Глава 8.
Одноприборные задачи с обратными критериями оптимизации

Представьте себе, что нам необходимо максимизировать суммарное запаздывание или максимизировать число запаздывающих требований, в отличие от классических задач, где эти значения нужно минимизировать. Несмотря на абсурдность такой максимизации, данные задачи представляют собой теоретический и практический интерес. В данном параграфе рассматриваются одноприборные задачи с "обратными" критериями оптимальности, например, задачи максимизации суммарного запаздывания и максимизации количества запаздывающих требований для одного прибора.

Формулируются эти задачи следующим образом, практически совпадающим с формулировками классических задач минимизации.

Необходимо обслужить n требований на одном приборе. Прерывания при обслуживании и обслуживание более одного требования в любой момент времени запрещены. Для каждого требования $j \in N = \{1, 2, \ldots, n\}$ заданы продолжительность обслуживания $p_j > 0$ и директивный срок его окончания d_j, где N – множество требований, которые необходимо обслужить. Прибор начинает обслуживание требований с момента времени 0. **Простои при обслуживании требований запрещены** (иначе задачи максимизации становятся тривиальными). Расписание обслуживания требований $\pi = (j_1, j_2, \ldots, j_n)$ строится с момента времени 0 и однозначно задаётся перестановкой элементов множества N. Обозначим через $C_{j_k}(\pi) = \sum\limits_{l=1}^{k} p_{j_l}$ время завершения обслуживания требований j_k при расписании π. Если заданы параметры $D_j,\ j = 1, 2, \ldots, n$, то при допустимом расписании $C_j \le D_j,\ j = 1, 2, \ldots, n$.

Требуется построить расписание π^* обслуживания требований множества N, при котором достигается максимум функции $F(\pi) = \sum\limits_{j=1}^{n} U_j(\pi)$. Обозначим данную задачу через $1(nd)||\max\sum U_j$. Аналогично, обозначается задача максимизации суммарного запаздывания $1(nd)||\max\sum T_j$ и другие задачи максимизации.

Запись (nd) означает, что рассматриваются расписания, при которых прибор не простаивает (non-delay), т.е. обслуживает все требования в интервале $[0, \sum p_j]$ без перерывов. В литературе можно встретить и другие задачи максимизации, обозначаемые символами $1(sa)|\ldots|\max\ldots$, в которых рассматриваются т.н. semi-active расписания.

В этом параграфе приводятся доказательства NP-трудности некоторых задач максимизации, а также полиномиальный графический алгоритм решения задачи $1(nd)||\max\sum T_j$.

С одной стороны, исследование данных задач само по себе является важной теоретической задачей. Алгоритмы решения данных задач могут быть использованы для вычисления верхних оценок, исследования свойств оптимальных расписаний, вычисления частичного порядка обслуживания требований для "исходных" задач, а также для сокращения перебора в алгоритмах решения "исходных" задач. Например, алгоритм решения задачи максимизации количества запаздывающих требований $1||\max\sum U_j$ используется для вычисления максимального количества запаздывающих требований, что, в свою очередь, позволяет сократить перебор в алгоритме решения задачи максимизации суммарного запаздывания $1||\max\sum T_j$. Значение $\max\sum T_j$ может быть использовано при решении задачи $1||(\alpha\sum E_j + \beta\sum T_j)$, т.е. если $\alpha > \beta$, тогда можно вычислить максимальное значение $\beta\sum T_j$ и после искать расписание, оптимальное только с точки зрения критерия $\sum E_j$.

Также теоретический интерес представляет корреляция между полиномиально разрешимыми и NP-трудными случаями для "исходной" и "обратной" задач.

С другой стороны, для данных задач существуют практические интерпретации и приложения. Например, монтажная команда должна смонтировать ветряные электрогенераторы (турбины) в разных районах страны. В каждом районе j необходимо смонтировать определен-

ное количество турбин. Время монтажа p_j зависит только от количества турбин и не зависит от погодных или климатических условий. Однако погода влияет на дополнительные расходы (например, на расход топлива, на зарплату рабочих и стоимость проживания рабочих, которая может быть выше зимой). Сумма этих дополнительных расходов начинает быстро снижаться после схода снега. Для каждого региона (т.е. для каждого требования) дан прогноз, когда ожидается сход снега, т.е. когда снег растает (этот момент времени можно интерпретировать как директивный срок). Целевая функция – минимизировать эти дополнительные расходы, т.е. целевая функция может быть интерпретирована как $\max \sum \max\{0, S_j - d'_j\}$, где $S_j = C_j - p_j$ и $d'_j = d_j - p_j$. В результате получили задачу $1||\max \sum T_j$.

Сведения о трудоёмкости задач максимизации сведены в таблицу 5.

Таблица 5. Сведения о трудоёмкости задач максимизации

Задача	Трудоёмкость и алгоритмы	Трудоёмкость соотв. задач минимизации
$1(nd)\|\|\max \sum U_j$	Полин. разрешима за время $O(n \log n)$	Полин. разрешима за время $O(n \log n)$
$1(nd)\|D_j\|\max \sum T_j$	NP-трудна	???
$1(nd)\|D_j\|\max \sum U_j$	NP-трудна	???
$1(nd)\|D_j\|\max \sum w_j C_j$	NP-трудна	???
$1(nd)\|D_j\|\max \sum C_j$	NP-трудна	???
$1(nd)\|\|\max \sum w_j U_j$	NP-трудна	NP-трудна
$1(nd)\|r_j\|\max \sum U_j$	NP-трудна	NP-трудна
$1(nd)\|r_j\|\max \sum w_j C_j$	NP-трудна	NP-трудна
$1(nd)\|r_j\|\max \sum C_j$	Полин. разрешима за время $O(n \log n)$	NP-трудна
$1(nd)\|prec, r_j\|\max C_{\max}$	$O(n^2)$	NP-трудна
$1(nd)\|prec, r_j\|\max f_{\max}$	$O(n^4)$	NP-трудна
$1(nd)\|r_j\|\max f_{\max}$	$O(n^3)$	NP-трудна
$1(nd)\|prec, r_j\|\max \sum C_j$	NP-трудна	NP-трудна
$1(nd)\|prec, r_j\|\max \sum U_j$	NP-трудна	NP-трудна
$1(nd)\|\|\max \sum w_j T_j$	NP-трудна. Алгоритм решения трудоёмкости $O(n \min\{\sum w_j, d_{\max}\})$	NP-трудна
$1(nd)\|r_j\|\max \sum w_j T_j$	NP-трудна	NP-трудна
$1(nd)\|r_j\|\max \sum T_j$	NP-трудна	NP-трудна
$1(nd)\|\|\max \sum T_j$	Алгоритм решения трудоёмкости $O(n \sum p_j)$. Алгоритм решения трудоёмкости $O(n^2)$	NP-трудна. Алгоритм решения трудоёмкости $O(n^4 \sum p_j)$
$1(sa)\|r_j\|\max \sum T_j$	Полин. разрешима за время $O(n^3)$	
$1(sa)\|r_j\|\max \sum w_j T_j$	NP-трудна	

Большинство результатов для задач максимизации, представленных в таблице, могут быть получены без особых усилий. Далее приведены три простых доказательства NP-трудности некоторых из перечисленных задач.

Лемма 34 *Задачи* $1(nd)|D_j|\max \sum T_j$ *и* $1(nd)|D_j|\max \sum U_j$ *NP-трудны.*

Доказательство. Доказательство проведем сведением ЗАДАЧИ РАЗБИЕНИЯ к данным задачам. Дан пример ЗАДАЧИ РАЗБИЕНИЯ с n числами, мы конструируем пример исследуемых задач следующим образом. В примере $n + 1$ требований, где $p_j = b_j$ и

$d_j = D_j = \sum\limits_{j=1}^{n} p_j + 1$, $j = 1, 2, \ldots, n$, т.е. при любом допустимом расписании эти требований не запаздывают. Пусть $p_{n+1} = 1$, $d_{n+1} = A$, $D_{n+1} = A + 1$.

Если ответ в примере ЗАДАЧИ РАЗБИЕНИЯ "ДА", тогда существует оптимальное расписание $\pi = (\pi_1, n+1, \pi_2)$, где $\{\pi_1\} = N'$, $\sum\limits_{j \in N'} p_j = A$, $\{\pi_2\} = N \setminus N'$ и $\sum\limits_{j=1}^{n+1} T_j(\pi) = 1$. Если ответ "НЕТ", тогда для всех допустимых расписаний $\sum\limits_{j=1}^{n+1} T_j = 0$. Соответственно, выполняются $\sum\limits_{j=1}^{n+1} U_j = 1$ и $\sum\limits_{j=1}^{n+1} U_j = 0$. $\square$

Лемма 35 *Задачи* $1(nd)|r_j|\max\sum w_jC_j$ *и* $1(nd)|D_j|\max\sum w_jC_j$ *NP-трудны.*

Доказательство. Доказательство проведем сведением ЗАДАЧИ РАЗБИЕНИЯ к данным задачам.Дан пример ЗАДАЧИ РАЗБИЕНИЯ с n числами, мы конструируем пример задачи $1(nd)|r_j|\max\sum w_jC_j$ следующим образом. В примере $n+1$ требований, где $p_j = w_j = b_j$ и $r_j = 0$, $j = 1, 2, \ldots, n$. Пусть $p_{n+1} = 1, w_{n+1} = 0$, $r_{n+1} = A$.

Если ответ в примере ЗАДАЧИ РАЗБИЕНИЯ "ДА", тогда существует оптимальное расписание $\pi = (\pi_1, n+1, \pi_2)$, где $\{\pi_1\} = N'$, $\sum\limits_{j \in \pi_1} p_j = A$, $\{\pi_2\} = N \setminus N'$, $\sum\limits_{j \in \pi_2} p_j = \sum\limits_{j \in N} b_j - A$ и

$$\sum_{j=1}^{n+1} w_jC_j(\pi) = \sum_{1 \le i \le j \le n} b_ib_j + \left(\sum_{j \in N} b_j - A\right),$$

т.к. при расписании $\pi' = (\pi_1, \pi_2, n+1)$ выполняется

$$\sum_{j=1}^{n+1} w_jC_j(\pi) = \sum_{1 \le i \le j \le n} b_ib_j.$$

Требования в частичных расписаниях π_1 и π_2 могут быть обслужены в любом порядке. Если ответ "НЕТ", то

$$\sum_{j=1}^{n+1} w_jC_j(\pi) < \sum_{1 \le i \le j \le n} b_ib_j + \left(\sum_{j \in N} b_j - A\right).$$

Поэтому задача $1(nd)|r_j|\max\sum w_jC_j$ NP-трудна.

Аналогично для задачи $1(nd)|D_j|\max\sum w_jC_j$, мы конструируем пример с множеством из $n+1$ требований, где $p_j = b_j, w_j = 0$, $D_j = \sum\limits_{i=1}^{n+1} p_i$, $j = 1, 2, \ldots, n$, а также $p_{n+1} = 1, w_{n+1} = 1$ и $D_{n+1} = A + 1$. Ответ для примера ЗАДАЧИ РАЗБИЕНИЯ "ДА" тогда и только тогда, когда $C_{n+1}(\pi) = A + 1$.

$\square$

Лемма 36 *Задача* $1(nd)|D_j|\max\sum C_j$ *NP-трудна.*

Доказательство. Очевидно, что две задачи $1|r_j|\min\sum C_j$ и $1|r_j|\min\sum S_j$ эквивалентны, т.к. $\sum\limits_{j \in N} C_j = \sum\limits_{j \in N} S_j + \sum\limits_{j \in N} p_j$. Известно, что задача $1|r_j|\min\sum C_j$ NP-трудна. Легко показать, что задачи $1(nd)|D_j|\max\sum C_j$ и $1|r_j|\min\sum S_j$ также эквивалентны. Пусть имеет пример задачи $1|r_j|\min\sum S_j$ с множеством N, содержащим n требований. Для примера задачи $1(nd)|D_j|\max\sum C_j$, зададим n требований с параметрами p'_j, D'_j, определенными по правилам: $p'_j = p_j$, $D'_j = \sum\limits_{j \in N} p_j - r_j$. Рассмотрим расписание $\pi = (j_1, j_2, \ldots, j_k, j_{k+1}, \ldots, j_n)$. Определим

$H_{j_k} = \sum\limits_{i=k+1}^{n} p_{j_i}$. Тогда задача максимизации $\sum\limits_{j\in N} C_j = n\sum\limits_{j\in N} p_j - \sum\limits_{j\in N} H_j$ сводится к задаче минимизации $\sum\limits_{j\in N} H_j$, т.е. сводится к задаче $1|r_j|\min\sum S_j$.

Пусть $\pi = (1, 2, \ldots, n)$ – оптимальное расписание для примера задачи $1|r_j|\min\sum S_j$. Тогда расписание $\pi' = (n, \ldots, 2, 1)$ будет оптимальным для соответствующего примера задачи $1|D_j|\max\sum C_j$.

Поэтому две задачи эквивалентны и, следовательно, задача $1(nd)|D_j|\max\sum C_j$ NP-трудна. □

8.1. Доказательство NP-трудности задачи $1(nd)||\max\sum w_jT_j$

Далеко не все доказательства NP-трудности для задач Теории Расписаний столь тривиальны. Для иллюстрации нетривиального доказательства рассмотрим задачу $1(nd)||\max\sum w_jT_j$. Доказательство проведем сведением ЗАДАЧИ РАЗБИЕНИЯ к частному случаю задачи $1(nd)||\max\sum w_jT_j$. Без потери общности, пусть в примере ЗАДАЧИ РАЗБИЕНИЯ $n > 3$ и $\sum\limits_{i=1}^{n} b_i > 10$.

Дан пример ЗАДАЧИ РАЗБИЕНИЯ, мы конструируем следующий пример задачи $1(nd)||\max\sum w_jT_j$:

$$\begin{cases} w_{2i} = M^i,\ i = 1, 2, \ldots, n, & (13.1)\\ w_{2i-1} = w_{2i} + b_i,\ i = 1, 2, \ldots, n, & (13.2)\\ p_{2i} = \sum\limits_{j=1}^{i-1} w_{2j} + \frac{1}{2}\sum\limits_{j\in N\setminus\{i\}} b_j,\ i = 1, 2, \ldots, n, & (13.3)\\ p_{2i-1} = p_{2i} + b_i,\ i = 1, 2, \ldots, n, & (13.4)\\ d_{2i} = d_{2i-1} = P - \sum\limits_{j=i}^{n} p_{2j},\ i = 1, 2, \ldots, n, & (13.5) \end{cases} \tag{13}$$

где $M = \left(n\sum\limits_{i=1}^{n} b_i\right)^{10}$ и $P = \sum\limits_{j=1}^{2n} p_j$.

Обозначим работы из множества $\bar{N} = \{1, 2, \ldots, 2n\}$ следующим образом:

$$V_1, V_2, V_3, V_4, \ldots, V_{2i-1}, V_{2i}, \ldots, V_{2n-1}, V_{2n}.$$

Каноническим расписанием будем называть расписание вида

$$(V_{n,1}, V_{n-1,1}, \ldots, V_{i,1}, \ldots, V_{1,1}, V_{1,2}, \ldots, V_{i,2}, \ldots, V_{n-1,2}, V_{n,2}),$$

где $\{V_{i,1}, V_{i,2}\} = \{V_{2i-1}, V_{2i}\},\ i = 1, 2, \ldots, n$.

Лемма 37 *Для каждого примера задачи $1(nd)||\max\sum w_jT_j$ существует оптимальное расписание вида $\pi = (G, H)$, где все требования $j \in H$ запаздывают, а все требования $i \in G$ не запаздывают. Все требования из множества G обслуживаются в порядке невозрастания значений $\frac{w_j}{p_j}$, а все требования из множества F обслуживаются в порядке неубывания значений $\frac{w_j}{p_j}$.*

Доказательство.

1) Предположим, что существует оптимальное расписание $\pi = (\pi_1, j, \pi_2, i, \pi_3)$, где все требования j запаздывают, а все требования i не запаздывают. Для расписания $\pi' = (\pi_1, i, j, \pi_2, \pi_3)$ выполняется: $F(\pi') - F(\pi) \geq w_j(T_j(\pi') - T_j(\pi)) + w_i(T_i(\pi') - T_i(\pi)) = w_j(p_i) + (0) > 0$. Получили противоречие, так как для расписания π' значение целевой функции больше, а

значит расписание π не оптимальное.

2) Рассмотрим оптимальное расписание вида $\pi = (G, H)$, где все требования $j \in H$ запаздывают, а все требования $i \in G$ не запаздывают. Покажем, что все требования $j \in H$ обслуживаются в порядке неубывания значений $\frac{w_j}{p_j}$. Предположим, что существует оптимальное расписание $\pi = (\pi_1, j_1, j_2, \pi_2)$, при котором требования j_1 и j_2 запаздывают и $\frac{w_{j_1}}{p_{j_1}} > \frac{w_{j_2}}{p_{j_2}}$, т.е. $w_{j_1}p_{j_2} > w_{j_2}p_{j_1}$. Для расписания $\pi' = (\pi_1, j_2, j_1, \pi_2)$, выполняется $F(\pi') - F(\pi) = w_{j_1}(T_{j_1}(\pi') - T_{j_1}(\pi)) + w_{j_2}(T_{j_2}(\pi') - T_{j_2}(\pi)) \geq w_{j_1}p_{j_2} - w_{j_2}\min\{p_{j_1}, T_{j_2}(\pi)\} > 0$. Получили противоречие, а значит, расписание $\pi = (\pi_1, j_1, j_2, \pi_2)$ не оптимальное.

3) Рассмотрим оптимальное расписание вида $\pi = (G, H)$, где все требования $j \in H$ запаздывают, а все требования $i \in G$ не запаздывают. Покажем что все требования $i \in G$ могут обслуживаться в порядке невозрастания значений $\frac{w_j}{p_j}$ при оптимальном расписании. Для всех требований $i \in G$ выполняется $d_i \geq \sum\limits_{k \in G} p_k$, иначе, если $d_i < \sum\limits_{k \in G} p_k$, то расписание $\pi' = (G \setminus \{i\}, i, H)$ "лучше" (т.е. для данного расписания значение целевой функции больше), а следовательно, получено противоречие. Поэтому все требования $i \in G$ могут быть обслужены в любом порядке, так как при любом порядке обслуживания все требования из G не запаздывают. $\square$

Следствием из этой леммы является следующее утверждение:

Лемма 38 *Для каждого примера задачи* $1(nd)||\max\sum T_j$ *существует оптимальное расписание вида* $\pi = (G, H) = (SPT, LPT)$*, где все требования* $j \in H$ *запаздывают, а все требования* $i \in G$ *не запаздывают. Все требования из множества* G *обслуживаются в порядке неубывания продолжительности обслуживания* (SPT)*, а все требования из множества* H *обслуживаются в порядке невозрастания продолжительности обслуживания* (LPT)*.*

Лемма 39 *Для частного случая* (13)*, все оптимальные расписания являются каноническими, или могут быть сведены к каноническим расписаниям, если упорядочить первые* n *требований в расписании по правилу* LPT (*longest processing time*)*.*

Доказательство.
1) Сначала докажем, что одно из двух требований, V_{2n} и V_{2n-1}, запаздывает при любом оптимальном расписании. Предположим, что оба требования не запаздывают при оптимальном расписании $\pi = (\pi_1, V_{2n}, \pi_2, V_{2n-1}, \pi_3, \pi_4)$, где запаздывают только требования из частичного расписания π_4. Мы рассматриваем только оптимальные расписания вида $\pi = (G, H)$, где все требования $j \in H$ запаздывают, а все требования $i \in G$ не запаздывают (см. лемму 37). Для расписания $\pi' = (\pi_1, \pi_2, V_{2n-1}, \pi_3, \pi_4, V_{2n})$ имеем:

$$\sum_{j=1}^{n} w_j T_j(\pi') - \sum_{j=1}^{n} w_j T_j(\pi) \geq w_{2n}T_{2n}(\pi') - p_{2n}\sum_{j \in \pi_4} w_j \geq$$

$$p_{2n}M^n - p_{2n}\sum_{i=1}^{n-1}(2M^i + b_i) = p_{2n}M^n - p_{2n}\left(2M\frac{M^{n-1}-1}{M-1} + \sum_{i=1}^{n-1} b_i\right) > 0.$$

Значит расписание π не оптимальное, т.е. одно из двух требований, V_{2n} и V_{2n-1}, запаздывает при любом оптимальном расписании.
2) Докажем верность следующего неравенства:

$$\frac{w_2}{p_2} < \frac{w_4}{p_4} < \cdots < \frac{w_{2n}}{p_{2n}}.$$

Необходимо доказать, что

$$\frac{M^{i-1}}{\sum\limits_{j=1}^{i-2} w_{2j} + \frac{1}{2}\sum\limits_{j\in N\setminus\{i-1\}} b_j} < \frac{M^{i}}{\sum\limits_{j=1}^{i-1} w_{2j} + \frac{1}{2}\sum\limits_{j\in N\setminus\{i\}} b_j}.$$

Обозначим $B_i = \frac{1}{2}\sum\limits_{j\in N\setminus\{i\}} b_j,\ i = 1, 2, \ldots, n$. Тогда мы имеем:

$$\frac{M^{i-1}}{M\frac{M^{i-2}-1}{M-1} + B_{i-1}} < \frac{M^{i}}{M\frac{M^{i-1}-1}{M-1} + B_{i}} \iff$$

$$\frac{1}{\frac{M(M^{i-2}-1)+B_{i-1}(M-1)}{M-1}} < \frac{M}{\frac{M(M^{i-1}-1)+B_{i}(M-1)}{M-1}} \iff$$

$$M(M^{i-1}-1) + B_i(M-1) < M[M(M^{i-2}-1) + B_{i-1}(M-1)] \iff$$

$$0 < M^2(B_{i-1}-1) - M(B_{i-1}+B_i-1) + B_i.$$

Последнее неравенство истинно, т.к. $M^2 > M \cdot 2\sum\limits_{j=1}^{n} b_j$ и $(B_{i-1}-1) > 1$. То есть исходное неравенство верно. Аналогично можно доказать, что

$$\frac{w_{2(i-1)-1}}{p_{2(i-1)-1}} < \frac{w_{2i}}{p_{2i}}, \quad \frac{w_{2(i-1)}}{p_{2(i-1)}} < \frac{w_{2i-1}}{p_{2i-1}} \quad \text{и} \quad \frac{w_{2(i-1)-1}}{p_{2(i-1)-1}} < \frac{w_{2i-1}}{p_{2i-1}}$$

для любого $i = 2, 3, \ldots, n$.

Тогда одно из двух требований, V_{2n} и V_{2n-1} является последним запаздывающим требованием при любом оптимальном расписании (см. лемму 37). В дальнейшем будем рассматривать только расписания вида $(V_{n,1}, \pi_\alpha, V_{n,2})$, где $\{V_{n,1}, V_{n,2}\} = \{V_{2n-1}, V_{2n}\}$.

3) Аналогично доказательствам из пп. 1) и 2), можно доказать, что для каждого $i = n-1$, $n-2, \ldots, 1$, одно из двух требований, V_{2i} и V_{2i-1} запаздывает при любом оптимальном расписании.

Поэтому лемма верна.

□

Теорема 14 *Задача* $1(nd)||\max\sum w_jT_j$ *NP-трудна (в обычном смысле).*

Доказательство. Для расписания $\pi = (V_{2n-1}, V_{2(n-1)-1}, \ldots, V_3, V_1, V_2, V_4, \ldots, V_{2(n-1)}, V_{2n})$ имеем следующее значение целевой функции:

$$F(\pi) = \sum_{j=1}^{n} w_{2j}T_{2j}(\pi) = \sum_{j=1}^{n} w_{2j}p_{2j}.$$

Рассмотрим каноническое расписание

$$\pi' = (V_{n,1}, V_{n-1,1}, \ldots, V_{i,1}, \ldots, V_{1,1}, V_{1,2}, \ldots, V_{i,2}, \ldots, V_{n-1,2}, V_{n,2}).$$

Определим переменную

$$x_i = \begin{cases} 1, & \text{если} \quad V_{i,2} = V_{2i-1}, \\ 0, & \text{если} \quad V_{i,2} = V_{2i}. \end{cases}$$

Тогда мы имеем:

$$\begin{aligned}
F(\pi') &= F(\pi) + \sum_{i=1}^{n} x_i \left[(w_{2i-1} - w_{2i}) \cdot T_{V_{2i}}(\pi) - (p_{2i-1} - p_{2i}) \left(\sum_{j=1}^{i-1} w_{V_{j,2}} \right) \right] = \\
& F(\pi) + \sum_{i=1}^{n} x_i \left[b_i \cdot p_{2i} - b_i \cdot \left(\sum_{j=1}^{i-1} w_{V_{j,2}} \right) \right] = \\
& F(\pi) + \sum_{i=1}^{n} x_i \left[b_i \cdot p_{2i} - b_i \cdot \left(\sum_{j=1}^{i-1} w_{2j} + \sum_{j=1}^{i-1} x_j b_j \right) \right] = \\
& F(\pi) + \sum_{i=1}^{n} x_i b_i \left[p_{2i} - \frac{1}{2} \sum_{j=i+1}^{n} x_j b_j - \sum_{j=1}^{i-1} w_{2j} - \frac{1}{2} \sum_{j=1}^{i-1} x_j b_j \right] = \\
& F(\pi) + \frac{1}{2} \sum_{i=1}^{n} x_i b_i \left(\sum_{j \in N \setminus \{i\}} b_j - \sum_{j \in N \setminus \{i\}} x_j b_j \right) = \\
& F(\pi) + \frac{1}{2} \sum_{i=1}^{n} \sum_{j \in N \setminus \{i\}} x_i \cdot b_i \cdot b_j \cdot (1 - x_j).
\end{aligned}$$

Когда существует подмножество $N' \subset N$ для примера ЗАДАЧИ РАЗБИЕНИЯ такое, что $\sum_{i \in N'} b_i = A$, тогда и только тогда при оптимальном каноническим расписании π^*, мы имеем

$$F(\pi^*) = F(\pi) + \frac{1}{2} A^2,$$

где $x_i = 1$, если $i \in N'$, и $x_j = 0$, если $iN \setminus N'$, т.к.

$$F(\pi^*) = F(\pi) + \frac{1}{2} \sum_{i \in N'} \sum_{j \in N \setminus N'} b_i \cdot b_j = F(\pi) + \frac{1}{2} A \cdot A.$$

Если ответ в примере ЗАДАЧИ РАЗБИЕНИЯ “НЕТ”, тогда

$$\begin{aligned}
F(\pi^*) &= F(\pi) + \frac{1}{2} \sum_{i=1}^{n} \sum_{j \in N \setminus \{i\}} x_i \cdot b_i \cdot b_j \cdot (1 - x_j) = \\
& F(\pi) + \frac{1}{2}(A - y)(A + y) = \\
& F(\pi) + \frac{1}{2} A^2 - \frac{1}{2} y^2,
\end{aligned}$$

где $y > 0$.

Теорема доказана.

□

Для задачи $1(nd)||\max \sum w_j T_j$ существует псевдополиномиальный алгоритм решения, поэтому задача NP-трудна в обычном смысле (не в сильном смысле).

8.2. Псевдополиномиальный алгоритм решения задачи $1(nd)||\max \sum T_j$

Алгоритм 1 основан на лемме 38, т.е. на том факте, что существует оптимальное расписание вида $\pi = (G, H) = (SPT, LPT)$, где все требования $j \in H$ запаздывают, а все требования $i \in G$ не запаздывают.

Algorithm 1 Алгоритм решения задачи $1(nd)||\max\sum T_j$.

1: Перенумеруем требования согласно правилу: $p_1 \geq p_2 \geq \cdots \geq p_n$. Если $p_i = p_{i+1}$, тогда $d_i \geq d_{i+1}$;
2: **for** $t := 0$ to $\sum_{i=2}^{n} p_i$ **do**
3: $\pi_1(t) := (1)$;
4: $f_1(t) := \max\{0, p_1 + t - d_1\}$;
5: **end for**
6: **for** $l := 2$ to n **do**
7: **for** $t := 0$ to $\sum_{i=l+1}^{n} p_i$ **do**
8: $\pi^1 := (j, \pi_{j-1}(t + p_j))$, $\pi^2 := (\pi_{j-1}(t), j)$;
9: $\Phi^1(t) := \max\{0, p_l + t - d_l\} + f_{l-1}(t + p_l)$;
10: $\Phi^2(t) := f_{l-1}(t) + \max\{0, \sum_{j=1}^{l} p_j + t - d_l\}$;
11: **if** $\Phi^1(t) < \Phi^2(t)$ **then**
12: $f_l(t) := \Phi^2(t)$;
13: $\pi_j(t) := \pi^2$;
14: **else**
15: $f_l(t) := \Phi^1(t)$;
16: $\pi_j(t) := \pi^1$;
17: **end if**
18: **end for**
19: **end for**

$\pi_n(0)$ – оптимальное расписание, а $f_n(0)$ – соответствующее оптимальное значение целевой функции (максимальное суммарное запаздывание).

Теорема 15 *Алгоритм 1 строит оптимальное расписание для задачи* $1(nd)||\max\sum T_j$ *за* $O(n\sum p_j)$ *операций.*

Доказательство. Доказательство от противного. Допустим, что существует оптимальное расписание $\pi^* = (G, H)$, где все требования $j \in H$ запаздывают, а все требования $i \in G$ не запаздывают. Требования из множества G обслуживаются в порядке SPT, а требования из множества H – в порядке LPT, для которого $f(\pi^*) > f(\pi_n(0)) = f_n(0)$.

Пусть $\pi' := \pi^*$. Для каждого $l = 1, 2, \ldots, n$ мы последовательно рассматриваем часть $\bar{\pi}_l \in \pi'$, $\{\bar{\pi}_l\} = \{1, \ldots, l\}$, расписания. Пусть $\pi' = (\pi_\alpha, \bar{\pi}_l, \pi_\beta)$. Если $\bar{\pi}_l \neq \pi_l(t)$, где $t = \sum_{i \in \{\pi_\alpha\}} p_i$, тогда примем $\pi' := (\pi_\alpha, \pi_l(t), \pi_\beta)$. Очевидно, что $f((\pi_\alpha, \bar{\pi}_l, \pi_\beta)) \leq f((\pi_\alpha, \pi_l(t), \pi_\beta))$. Аналогичную операцию проделываем для каждого $l = 1, 2, \ldots, n$. В конце получаем $f(\pi^*) \leq f(\pi') \leq f_n(0)$. Поэтому расписание $\pi_n(0)$ так же оптимально.

Очевидно, что трудоёмкость алгоритма составляет $O(n\sum p_j)$ операций, так как в цикле 6–19 выполняется $n - 1$ итераций и просматриваются целочисленные точки, входящие в интервал $\left[0, \sum_{j=1}^{n} p_j\right]$. □

8.3. Графический алгоритм решения задачи $1(nd)||\max\sum T_j$

В этом параграфе представлен Графический Алгоритм (GrA) решения задачи $1(nd)||\max\sum T_j$, основанный на уже известной нам идее модификации алгоритма динамического программирования. GrA представляет собой модификацию алгоритма 1, в которой функция $f_l(t)$ определена для любого значения $t \in (-\infty, +\infty)$ (не только для целых t), но мы

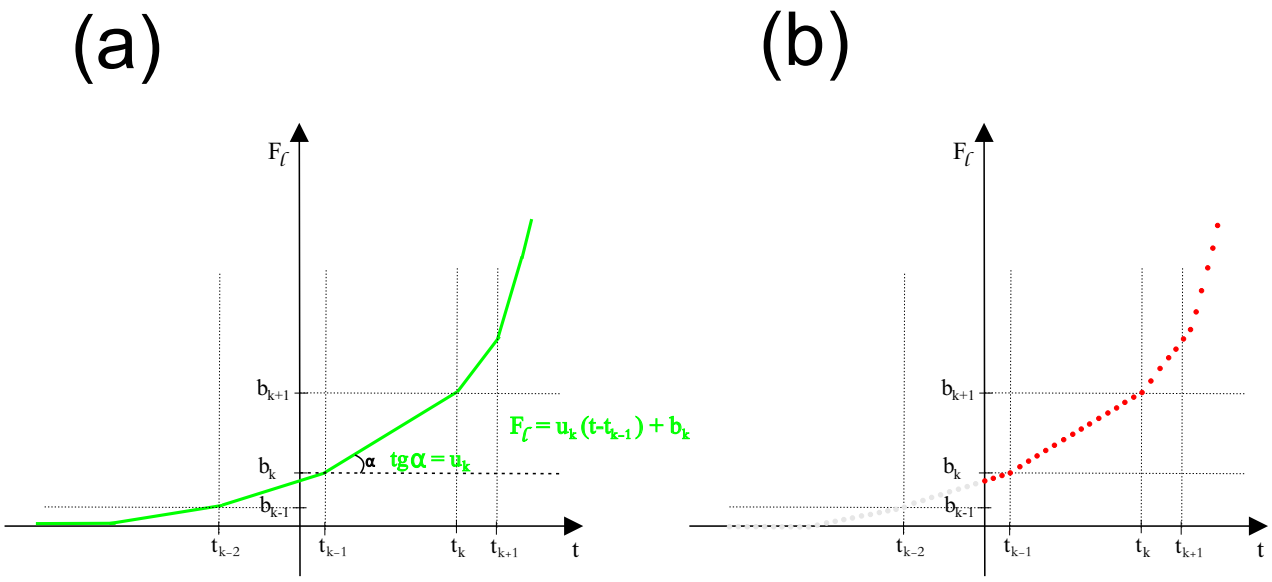


Рис. 7. Функция $f_l(t)$ в алгоритме 1 и в алгоритме GrA

вычисляем ее значения только в *точках излома*. Как будет показано далее, количество таких точек полиномиально. В этом параграфе помимо самого алгоритма GrA проиллюстрирована работа алгоритма на конкретном примере.

На каждом шаге алгоритма GrA, мы сохраняем функцию $f_l(t)$ в табличном виде, как представлено в табл. 6.

Таблица 6. Функция $f_l(t)$

t	$(-\infty, t_1]$	$(t_1, t_2]$	...	$(t_m, +\infty)$
$f_l(t)$	$b_1 = 0$	b_2	...	b_{m+1}
количество запаздывающих требований	$u_1 = 0$	u_2	...	u_{m+1}
частичное оптимальное расписание	π_1	π_2	...	π_{m+1}

Заметим, что значения π_l из табл. 6 и $\pi_l(t)$ из алгоритма 1 имеют разный смысл.

Записи в таблице означают следующее. Для каждого значения $t \in (t_{k-1}, t_k]$, оптимальным будет частичное расписание π_k, в котором u_k запаздывающих требований, и которому соответствует значение целевой функции $f_l(t) = u_k \cdot (t - t_{k-1}) + b_k$ (см. рис. 7). Функция $f_l(t)$ определена не только для целых t, но и для вещественных t.

Для упрощения описания алгоритма GrA мы рассматриваем всю ось t, т.е. $t \in (-\infty, +\infty)$. В теореме 16 показано, что эта таблица соответствует непрерывной, кусочно-линейной выпуклой функции $f_l(t)$. Точки $t_1, t_2, \ldots, t_m$ называются *точками излома*. Только в них происходит изменение (увеличение) количества запаздывающих требований с u_{k-1} на u_k (это означает, что меняется наклон кусочно-линейной функции). Заметим, что точки t_k могут быть нецелочисленными. Для описания каждого линейного сегмента функции, мы храним его наклон u_k и значение функции b_k в точке $t = t_{k-1}$.

В GrA функция $f_l(t)$ соответствует тем же рекурсивным уравнениям Беллмана, что и функция $f_l(t)$ в алгоритме 1, т.е. для каждого $t \in Z \bigcap \left[0, \sum_{j=2}^{n} p_j\right]$, $f_l(t)$ имеет тоже значение, что и в алгоритме 1 (см. рис.7), но теперь функция определена на всем интервале $t \in (-\infty, +\infty)$. В результате, "состояния" t, соответствующие одному и тому же оптимальному частичному расписанию, сгруппированы в интервалы. На рис. 7 (a), показана функция $f_l(t)$ из GrA, а на рис. 7 (b) – функция $f_l(t)$ из алгоритма 1.

Теперь опишем алгоритм GrA.

Графический алгоритм GrA

Шаг 1. Перенумеруем требования согласно правилу: $p_1 \geq p_2 \geq \cdots \geq p_n$. Если $p_i = p_{i+1}$, тогда $d_i \geq d_{i+1}$;

Шаг 2. $l := 1,\ \pi_1(t) := (1)$, функция $f_1(t) := \max\{0, p_1 + t - d_1\}$ определена для всех t. Функция $f_1(t)$ задается таблицей 7.

Таблица 7. Функция $f_1(t)$

t	$(-\infty, d_1 - p_1]$	$(d_1 - p_1, +\infty)$
$f_1(t)$	0	0
количество запаздывающих требований	0	1
частичное оптимальное расписание	(1)	(1)

Шаг 3. Пусть $l > 1$, а функция $f_{l-1}(t)$ вычислена (см. табл. 8).

Таблица 8. Функция $f_{l-1}(t)$

t	$(-\infty, t_1]$	$(t_1, t_2]$	$\ldots$	$(t_m, +\infty)$
$f_{l-1}(t)$	$b_1 = 0$	b_2	$\ldots$	b_{m+1}
количество запаздывающих требований	$u_1 = 0$	u_2	$\ldots$	u_{m+1}
частичное оптимальное расписание	π_1	π_2	$\ldots$	π_{m+1}

Из функции $f_{l-1}(t)$ построим функцию $f_l(t)$. Промежуточные функции $\Phi^1(t)$ и $\Phi^2(t)$ также хранятся в виде табл. 6.

Шаг 3.1. Функция $\Phi^1(t)$ строится из функции $f_{l-1}(t)$ при помощи следующих операций. Мы сдвигаем график функции $f_{l-1}(t)$ влево на p_l единиц и в таблицу, соответствующую "сдвинутой" функции $f_{l-1}(t)$, добавляем новую колонку, соответствующую новой точке излома $t' = d_l - p_l$. Если $t_k - p_l < t' < t_{k+1} - p_l,\ k+1 \leq m$, тогда в таблицу $\Phi^1(t)$ мы добавляем два интервала: $(t_k - p_l, t']$ и $(t', t_{k+1} - p_l]$. Далее мы увеличиваем все значения $u_{k+1}, u_{k+2}, \ldots, u_{m+1}$ на 1, т.е. количество запаздывающих требований (и наклон соотв. линейного сегмента графика) возрастает. Соответствующее частичное расписание π^1 строится добавлением требования l в начало предыдущего частичного расписания. Таким образом получена табл. 9, соответствующая функции $\Phi^1(t)$.

Шаг 3.2. Функция $\Phi^2(t)$ строится из функции $f_{l-1}(t)$ при помощи следующих операций. В таблицу, соответствующую функции $f_{l-1}(t)$, добавляем новую колонку, соответствующую новой точке излома $t' = d_l - \sum_{i=1}^{l} p_i$. Если $t_h < t' < t_{h+1},\ h+1 \leq m$, тогда в таблицу $\Phi^2(t)$ мы добавляем два интервала: $(t_h, t']$ и $(t', t_{h+1}]$. Далее мы увеличиваем все значения $u_{h+1}, u_{h+2}, \ldots, u_{m+1}$ на 1, т.е. количество запаздывающих требований (и наклон соотв. линейного сегмента графика) возрастает. Соответствующее частичное расписание π^2 строится добавлением требования l в конец предыдущего частичного расписания. Таким образом получена табл. 10, соответствующая функции $\Phi^2(t)$.

Шаг 3.3. Теперь мы строим таблицу, соответствующую функции

$$f_l(t) = \max\{\Phi^1(t), \Phi^2(t)\}.$$

В интервалах, образованных точками из двух таблиц 9 и 10 мы сравниваем значения функций $\Phi^1(t)$ и $\Phi^2(t)$ и ищем точки пересечения их графиков. Этот шаг требует порядка $O(m)$ операций. Если в итоговой таблице, соответствующей функции $f_l(t)$, встречается ситуация как в табл. 12, мы удаляем колонку $(t_k, t_{k+1}]$ и объединяем оба интервала, т.е. принимаем $t_k := t_{k+1}$.

В теореме 16 доказано, что для каждого l, число колонок в таблице $f_l(t)$ не больше $l + 1 \leq n + 1$.

Шаг 3.4. Если $l = n$, тогда переходим к шагу 4 иначе $l := l + 1$ и переходим к шагу 3.

Таблица 9. Функция $\Phi^1(t)$

t	$(-\infty, t_1 - p_l]$	$(t_1 - p_l, t_2 - p_l]$	$\ldots$	$(t_k - p_l, t']$	$(t', t_{k+1} - p_l]$	$(t_{k+1} - p_l, t_{k+2} - p_l]$	$\ldots$	$(t_m - p_l, +\infty)$
$\Phi^1(t)$	$b_1 = 0$	b_2	$\ldots$	b_{k+1}	b'	b'_{k+2}	$\ldots$	b'_{m+1}
количество запаздывающих требований	$u_1 = 0$	u_2	$\ldots$	u_{k+1}	$u_{k+1} + 1$	$u_{k+2} + 1$	$\ldots$	$u_{m+1} + 1$
частично оптимальное расписание π^1	(l, π_1)	(l, π_2)	$\ldots$	(l, π_{k+1})	(l, π_{k+1})	(l, π_{k+2})	$\ldots$	(l, π_{m+1})

$b' = b_{k+1} + (t' - (t_k - p_l)) \cdot u_{k+1}$, $b'_{k+2} = b' + ((t_{k+1} - p_l) - t') \cdot (u_{k+1} + 1)$, $\ldots$, $b'_{m+1} = b_m{}' + ((t_m - t_{m-1}) \cdot (u_m + 1)$

Таблица 10. Функция $\Phi^2(t)$

t	$(-\infty, t_1]$	$(t_1, t_2]$	$\ldots$	$(t_h, t']$	$(t', t_{h+1}]$	$(t_{h+1}, t_{h+2}]$	$\ldots$	$(t_m, +\infty)$
$\Phi^2(t)$	$b_1 = 0$	b_2	$\ldots$	b_{h+1}	b'	b'_{h+2}	$\ldots$	b'_{m+1}
количество запаздывающих требований	$u_1 = 0$	u_2	$\ldots$	u_{h+1}	$u_{h+1} + 1$	$u_{h+2} + 1$	$\ldots$	$u_{m+1} + 1$
частично оптимальное расписание π^2	(π_1, l)	(π_2, l)	$\ldots$	(π_{h+1}, l)	(π_{h+1}, l)	(π_{h+2}, l)	$\ldots$	(π_{m+1}, l)

$b' = b_{h+1} + (t' - t_h) \cdot u_{h+1}$, $b'_{h+2} = b' + (t_{h+1} - t') \cdot (u_{h+1} + 1)$, $\ldots$, $b'_{m+1} = b_m{}' + (t_m - t_{m-1}) \cdot (u_m + 1)$

Таблица 11. Функция $f_l(t)$

t	$(-\infty, t_1]$	$(t_1, t_2]$	$\ldots$	$(t_{k-1}, t_k]$	$(t_k, t_{k+1}]$	$\ldots$	$(t_m, +\infty)$
$f_l(t)$	$b_1 = 0$	b_2	$\ldots$	b_k	b_{k+1}	$\ldots$	b_{m+1}
количество запаздывающих требований	$u_1 = 0$	u_2	$\ldots$	u_k	u_{k+1}	$\ldots$	u_{m+1}
частично оптимальное расписание	π_1	π_2	$\ldots$	π_k	π_{k+1}	$\ldots$	π_{m+1}

Шаг 4. В таблице $f_n(t)$, мы находим колонку $(t_k, t_{k+1}]$, где $t_k < 0 \leq t_{k+1}$. Тогда мы имеем оптимальное расписание $\pi^* = \pi_{k+1}$ и оптимальное значение целевой функции $F(\pi^*) = = b_{k+1} + (0 - t_k) \cdot u_{k+1}$.

Очевидно, что GrA строит то же решение, что и алгоритм 1. В то время, как в алгоритме 1 рассматриваются все целочисленные точки $t \in \left[0, \sum_{j=2}^{n} p_j\right]$, в GrA рассматриваются только точки излома, определенные на всей оси t, такие, что все значения $f_l(t)$, $l \in \{1, 2, \ldots, n\}$, $t \in Z \cap \left[0, \sum_{j=l+1}^{n} p_j\right]$ соответствует тем же значениям, определенным в алгоритме 1. Поэтому алгоритм GrA строит точное решение.

Теорема 16 *Трудоёмкость алгоритма GrA составляет $O(n^2)$ операций.*

Доказательство.

Покажем, что все функции $f_l(t)$, $l = 1, 2, \ldots, n$, являются непрерывными кусочно-линейными выпуклыми функциями.

Очевидно, что функция $f_1(t)$ – непрерывная кусочно-линейная выпуклая функция с одной точкой излома. Согласно шагу 3 алгоритма обе функции $\Phi^1(t)$ и $\Phi^2(t)$ также являются непрерывными, кусочно-линейными и выпуклыми. Поэтому функция

$$f_2(t) = \max\{\Phi^1(t), \Phi^2(t)\}$$

также непрерывная, кусочно-линейная и выпуклая.

Продолжая данные рассуждения для других l, получим, что все функции $f_l(t), l = 1, 2, \ldots, n$, обладают этим характеристиками.

Теперь предположим, что на шаге 3 алгоритма мы получили табл. 11 для некоторой функции $f_l(t)$. Покажем, что выполняется неравенство $u_1 < u_2 < \cdots < u_k < u_{k+1} < \cdots < u_{m+1}$. Предположим, что $u_k > u_{k+1}$. Тогда для каждого $t \in (t_k, t_{k+1}]$, где t – время начала обслуживания требований из частичного расписания (π_k или π_{k+1}), имеем $F(\pi_k) > F(\pi_{k+1})$ т.к. выполняется $u_k > u_{k+1}$. То есть получили противоречие. Напомним, что функция $f_l(t)$ является непрерывной кусочно-линейной выпуклой функцией, и $b_{k+1} = b_k + (t_k - t_{k-1}) \cdot u_k$.

Согласно алгоритму GrA, если $u_k = u_{k+1}$, тогда мы удаляем колонку, соответствующую точке излома u_{k+1} и объединяем оба интервала.

Таблица 12. Удаление колонки

t	...	$(t_{k-1}, t_k]$	$(t_k, t_{k+1}]$	...
$f_l(t)$	...	...	...	...
количество запаздывающих требований	...	u_k	$u_{k+1} = u_k$	...
частичное оптимальное расписание π_l	...	...	...	...

Заметим, что данный факт можно доказать иначе. Функции $\Phi^1(t)$ и $\Phi^2(t)$ выпуклы, а следовательно функция

$$f_l(t) = \max\{\Phi^1(t), \Phi^2(t)\}$$

также выпукла. Значения u_k соответствуют $tg\alpha$, где α угол между осью t и линейным сегментом функции $f_l(t)$.

Мы имеем $u_1 < u_2 < \cdots < u_k < u_{k+1} < \cdots < u_{m+1}$, где u_i, $i = 1, 2, \ldots, m+1$, – количество запаздывающих требований. Тогда $m + 1 \leq l + 1 \leq n + 1$. Как следствие мы имеем не более l точек излома, что значит, не более $l + 1 \leq n + 1$ колонок для каждой функции $f_l(t)$, $l = 1, 2, \ldots, n$.

Поэтому трудоёмкость Шага 3 алгоритма составляет $O(n)$ операций для каждого $l = 1, 2, \ldots, n$. Значит трудоёмкость алгоритма GrA составляет $O(n^2)$ операций.

□

Стоит заметить, что уже для задачи $1(nd)||\max\sum w_jT_j$ аналогичный алгоритм имеет трудоёмкость $O(\min\{2^n, n\cdot\min\{d_{\max}, \sum w_j\}\})$ операций.

8.4. Иллюстрация работы Графического Алгоритма на примере

Рассмотрим пример задачи $1||\max\sum T_j$, входные данные которого представлены в табл. 13.

Таблица 13. Входные данные

j	1	2	3	4
p_j	30	22	12	5
d_j	32	35	38	40

Опишем таблицы, сохраняемые на каждой итерации l, алгоритма GrA.
Пусть $l = 1$**.** Сохраняем таблицу 14.

Таблица 14. Функция $f_1(t)$

t	$(-\infty, 2]$	$(2, +\infty)$
$f_1(t)$	0	0
u	0	1
π_1	(1)	(1)

График функции $f_1(t)$ представлен на рис. 8 (а).
Пусть $l = 2$**.** Получаем следующие результаты. Вычисляем новую точку излома $t' = d_2 - p_2 = 13$, и табл. 15 для функции $\Phi^1(t)$.

Таблица 15. Функция $\Phi^1(t)$

t	$(-\infty, -20]$	$(-20, 13]$	$(13, +\infty)$
$\Phi^1(t)$	0	0	33
u	0	1	2
π^1	(2,1)	(2,1)	(2,1)

Вычисляем новую точку излома $t' = d_2 - (p_1 + p_2) = -17$, и табл. 16 для функции $\Phi^2(t)$.

Таблица 16. Функция $\Phi^2(t)$

t	$(-\infty, -17]$	$(-17, 2]$	$(2, +\infty)$
$\Phi^2(t)$	0	0	19
u	0	1	2
π^2	(1,2)	(1,2)	(1,2)

Теперь проверим каждый из интервалов $(-\infty, -20]$, $(-20, -17]$, $(-17, 2]$, $(2, 13]$, $(13, +\infty)$, и проверим, не пересекаются ли функции $\Phi^1(t)$ и $\Phi^2(t)$ на этих интервалах. Для интервала $t \in (2, 13]$ из неравенства $\Phi^1(t) = (t{+}20){\cdot}1{+}0 = \Phi^2(t) = (t{-}2){\cdot}2{+}19$ получаем точку пересечения $(5, 25)$. Комбинируя результаты из таблиц $\Phi^1(t)$ и $\Phi^2(t)$, получаем табл. 17 функции $f_2(t)$. Заметим, что для $t \leq 5$, частичное расписание (2,1), соответствующее функции $\Phi^1(t)$ является оптимальным, в то время, как для $t > 5$ оптимальным является частичное расписание $(1, 2)$, соответствующее функции $\Phi^2(t)$.

Таблица 17. Функция $f_2(t)$

t	$(-\infty, -20]$	$(-20, 5]$	$(5, +\infty)$
$f_2(t)$	0	0	25
u	0	1	2
π_2	(2,1)	(2,1)	(1,2)

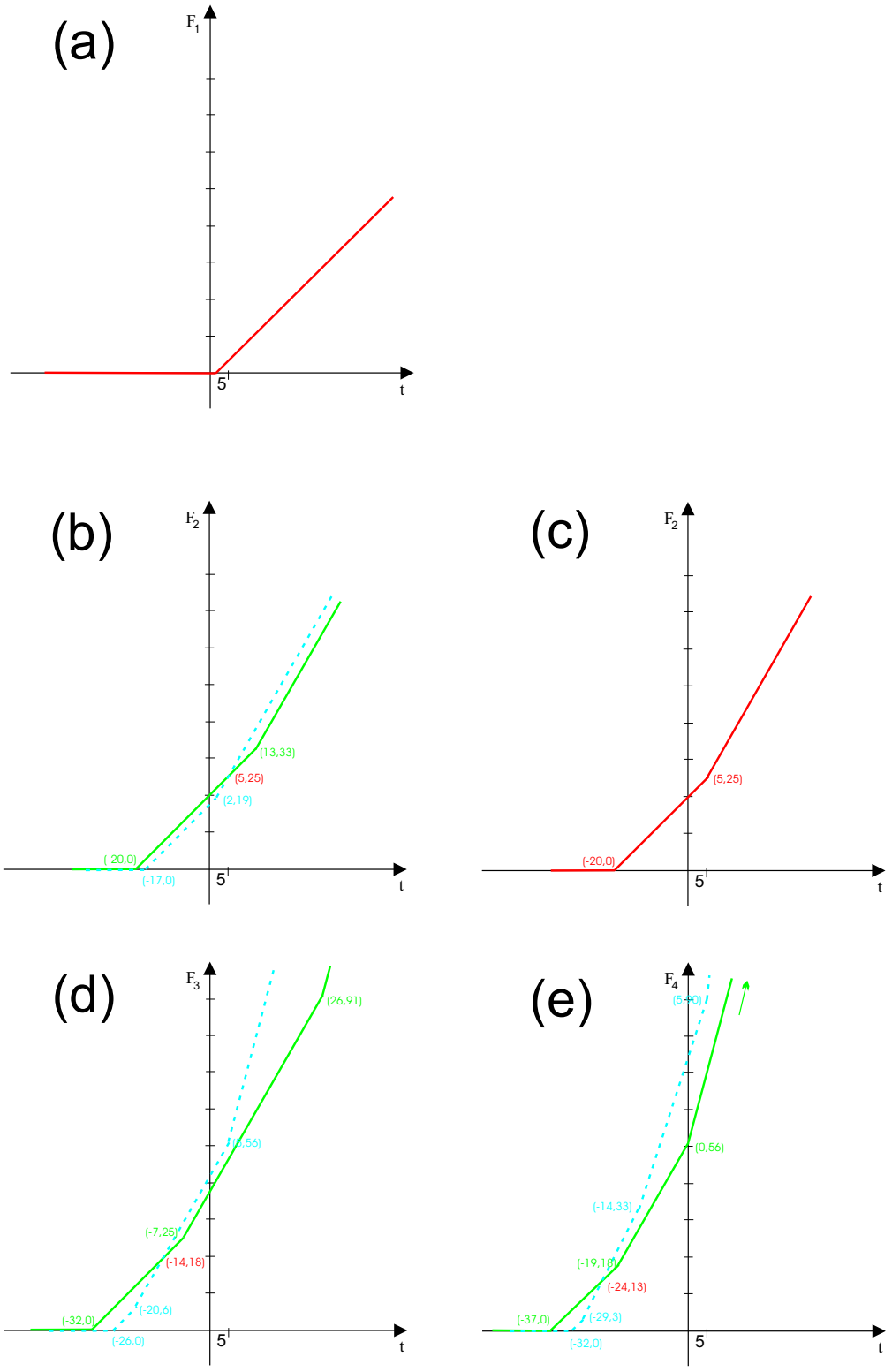


Рис. 8. Пример

График функции $f_1(t)$ представлен на рис. 8 (c). Он получен из графиков, представленных на рис. 8 (b). Аналогично рассмотрим оставшиеся итерации $l = 3, 4$.
Пусть $l = 3$. Получаем следующие результаты.
Вычисляем новую точку излома $t' = d_3 - p_3 = 26$, и табл. 18 для функции $\Phi^1(t)$.

Таблица 18. Функция $\Phi^1(t)$ для $l = 3$

t	$(-\infty, -32]$	$(-32, -7]$	$(-7, 26]$	$(26, +\infty)$
$\Phi^1(t)$	0	0	25	91
u	0	1	2	3
π^1	(3,2,1)	(3,2,1)	(3,1,2)	(3,1,2)

Вычисляем новую точку излома $t' = d_3 - (p_1 + p_2 + p_3) = -26$, и табл. 19 для функции $\Phi^2(t)$.

Таблица 19. Функция $\Phi^2(t)$ для $l = 3$

t	$(-\infty, -26]$	$(-26, -20]$	$(-20, 5]$	$(5, +\infty)$
$\Phi^2(t)$	0	0	6	56
u	0	1	2	3
π^2	(2,1,3)	(2,1,3)	(2,1,3)	(1,2,3)

Точка пересечения двух графиков функций $\Phi^1(t)$ и $\Phi^2(t)$ – $(-14, 18)$. Функция $f_3(t)$ представлена в табл. 20. Ее график показан на рис. 8 (d).

Таблица 20. Функция $f_3(t)$

t	$(-\infty, -32]$	$(-32, -14]$	$(-14, 5]$	$(5, +\infty)$
$f_3(t)$	0	0	18	56
u	0	1	2	3
π_3	(3,2,1)	(3,2,1)	(2,1,3)	(1,2,3)

Пусть $l = 4$**.** Получаем следующие результаты.
Вычисляем новую точку излома $t' = d_4 - p_4 = 35$ и табл. 21 для функции $\Phi^1(t)$.

Таблица 21. Функция $\Phi^1(t)$ для $l = 4$

t	$(-\infty, -37]$	$(-37, -19]$	$(-19, 0]$	$(0, 35]$	$(35, +\infty)$
$\Phi^1(t)$	0	0	18	56	161
u	0	1	2	3	4
π^1	(4,3,2,1)	(4,3,2,1)	(4,2,1,3)	(4,1,2,3)	(4,1,2,3)

Вычисляем новую точку излома $t' = d_4 - (p_1 + p_2 + p_3 + p_4) = -29$ и табл. 22 для функции $\Phi^2(t)$.

Таблица 22. Функция $\Phi^2(t)$ для $l = 4$

t	$(-\infty, -32]$	$(-32, -29]$	$(-29, -14]$	$(-14, 5]$	$(5, +\infty)$
$\Phi^2(t)$	0	0	3	33	90
u	0	1	2	3	4
π^2	(3,2,1,4)	(3,2,1,4)	(3,2,1,4)	(2,1,3,4)	(1,2,3,4)

Точка пересечения двух графиков функций $\Phi^1(t)$ и $\Phi^2(t)$ – $(-24, 13)$. Функция $f_4(t)$ представлена в табл. 23. Ее график показан на рис. 8 (e).

Таблица 23. Функция $f_4(t)$

t	$(-\infty, -37]$	$(-37, -24]$	$(-24, -14]$	$(-14, 5]$	$(5, +\infty)$
$f_4(t)$	0	0	13	33	90
u	0	1	2	3	4
π_4	(4,3,2,1)	(4,3,2,1)	(3,2,1,4)	(2,1,3,4)	(1,2,3,4)

Мы получили оптимальное значение целевой функции $f_4(0) = 33 + 14 \cdot 3 = 75$, которому соответствует оптимальное расписание $\pi_4(0) = (2, 1, 3, 4)$.

Глава 9.
Задачи с одним невозобновимым ресурсом

В данной главе рассматриваются одноприборные задачи с невозобновимым ресурсом. Этот ресурс необходим для обслуживания требований, причем потребности в ресурсе у требований может отличаться. В качестве такого ресурса могут выступать, например, деньги, топливо, прочие расходные материалы. Так как подобные задачи часто возникают в бюджетных организациях, для которых ресурсом являются деньги, эти задачи также называются *финансовыми*.

Постановка этих задач совпадает с классической постановкой одноприборных задач. Дополнительно заданы невозобновимый ресурс G (деньги, топливо и т.п.) и моменты времени поступления ресурса $\{t_0, t_1, \dots, t_y\},\ t_0 = 0,\ t_0 < t_1 < \dots < t_y$. В каждый момент времени $t_i,\ i = 0, 1, \dots, y$, поступает $G(t_i) \geq 0$ единиц ресурса. Для каждого требования $j \in N$, задано потребление ресурса $g_j \geq 0$, которое происходит в момент начала обслуживания. Поэтому выполняется равенство

$$\sum_{j=1}^{n} g_j = \sum_{i=0}^{y} G(t_i).$$

Обозначим через S_j время начала обслуживания требования j. Расписание $S = (S_{j_1}, S_{j_2}, \dots, S_{j_n})$ описывает в том числе порядок обслуживания требований: $\pi = (j_1, j_2, \dots, j_n)$. Расписание $S = (S_{j_1}, S_{j_2}, \dots, S_{j_n})$ является допустимым, если выполняется, помимо классических условий, следующее неравенство:

$$\sum_{k=1}^{i} g_{j_k} \leq \sum_{\forall l:\ t_l \leq S_{j_i}} G(t_l).$$

Перестановку π также называют расписанием, т.к. по этой перестановке можно вычислить расписание $S = (S_{j_1}, S_{j_2}, \dots, S_{j_n})$ за время $O(n)$.

Такие одноприборные задачи обозначают $1|NR, \dots| \dots$, где NR обозначает присутствие в условии задачи невозобновимого ресурса (non-renewable).

В табл. 24 представлены некоторые сведения о трудоёмкости одноприборных задач с невозобновимым ресурсом.

Таблица 24. Трудоёмкость задач

Цел. функц.	Частный случай	Трудоёмкость
$C_{\max}$		NP-трудна в сильном смысле. Сведение ЗАДАЧИ 3-РАЗБИЕНИЯ
$\sum U_j$		NP-трудна в сильном смысле. Сведение ЗАДАЧИ 3-РАЗБИЕНИЯ
$\sum C_j$		NP-трудна в сильном смысле. Сведение $1\|r_j\|\sum C_j$
$L_{\max}$		NP-трудна в сильном смысле. Сведение ЗАДАЧИ 3-РАЗБИЕНИЯ
$\sum T_j$	$d_j = d$	NP-трудна в сильном смысле. Сведение ЗАДАЧИ 3-РАЗБИЕНИЯ
$\sum T_j$	$g_j = g$	NP-трудна (задача $1\|\|\sum T_j$ является частным случаем)
$\sum T_j$	$p_j = p$	NP-трудна. Сведение ЗАДАЧИ РАЗБИЕНИЯ.
$\sum T_j$	$d_j = d,\ g_j = g$	NP-трудна. Сведение ЗАДАЧИ РАЗБИЕНИЯ.
$\sum T_j$	$p_j = p,$ $g_1 \leq g_2 \leq \dots \leq g_n,$ $d_1 \leq d_2 \leq \dots \leq d_n$	полиномиально разрешима, оптимальное расписание: $\pi^* = (1, 2, \dots, n)$

В качестве иллюстрации приведем несложное доказательство NP-трудности некоторых из перечисленных задач.

ЗАДАЧА 3-РАЗБИЕНИЯ. Задано множество $N = \{b_1, b_2, \ldots, b_n\}$ чисел, где $n = 3m$, $\sum_{i=1}^{n} b_j = mB$ и $\frac{B}{4} \leq b_j \leq \frac{B}{2}$, $j = 1, 2, \ldots, n$. Необходимо ответить на вопрос, существует ли такое разбиение множества N на m подмножеств $N_1, N_2, \ldots, N_m$, каждое из которых содержит ровно три числа, и суммы чисел которых равны, т.е.

$$\sum_{b_j \in N_1} b_j = \sum_{b_j \in N_2} b_j = \cdots = \sum_{b_j \in N_m} b_j = B?$$

Известно, что ЗАДАЧА 3-РАЗБИЕНИЯ является NP-трудной в сильном смысле.

Теорема 17 *Задачи* $1|NR|C_{\max}$, $1|NR, d_j = d|\sum T_j$, $1|NR|\sum U_j$ *и* $1|NR|L_{\max}$ – NP*-трудны в сильном смысле.*

Доказательство.

Доказательство проведем сведением ЗАДАЧИ 3-РАЗБИЕНИЯ к частному случаю задач.

Рассмотрим задачу $1|NR|C_{\max}$. Дан пример ЗАДАЧИ 3-РАЗБИЕНИЯ, построим пример задачи $1|NR|C_{\max}$ следующим образом. В пример зададим n требований с параметрами $g_j = p_j = b_j$, $j = 1, 2, \ldots, n$. Времена поступления ресурса определены как $\{t_0, t_1, \ldots, t_{m-1}\} = \{0, B, 2B, \ldots, (m-1)B\}$ и $G(t_0) = G(t_1) = \cdots = G(t_{m-1}) = B$. Очевидно, что $C_{\max} = mB$ тогда и только тогда, когда ответ в примере ЗАДАЧИ 3-РАЗБИЕНИЯ “ДА”. В этом случае в оптимальном расписании нет простоев прибора.

Для задач $1|NR, d_j = d|\sum T_j$, $1|NR|\sum U_j$ и $1|NR|L_{\max}$ дополнительно определим $d_j = d = mB$ для $j = 1, 2, \ldots, n$. Тогда $\sum T_j = 0$ выполняется тогда и только тогда, когда ответ в примере ЗАДАЧИ 3-РАЗБИЕНИЯ “ДА”. Аналогично $\sum U_j = 0$ и $L_{\max} = 0$ выполняются тогда и только тогда, когда ответ в примере ЗАДАЧИ 3-РАЗБИЕНИЯ “ДА”.

□

Глава 10.
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 $(p_1, p_2, \ldots, p_n, d_1, d_2, \ldots, d_n)$. 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.

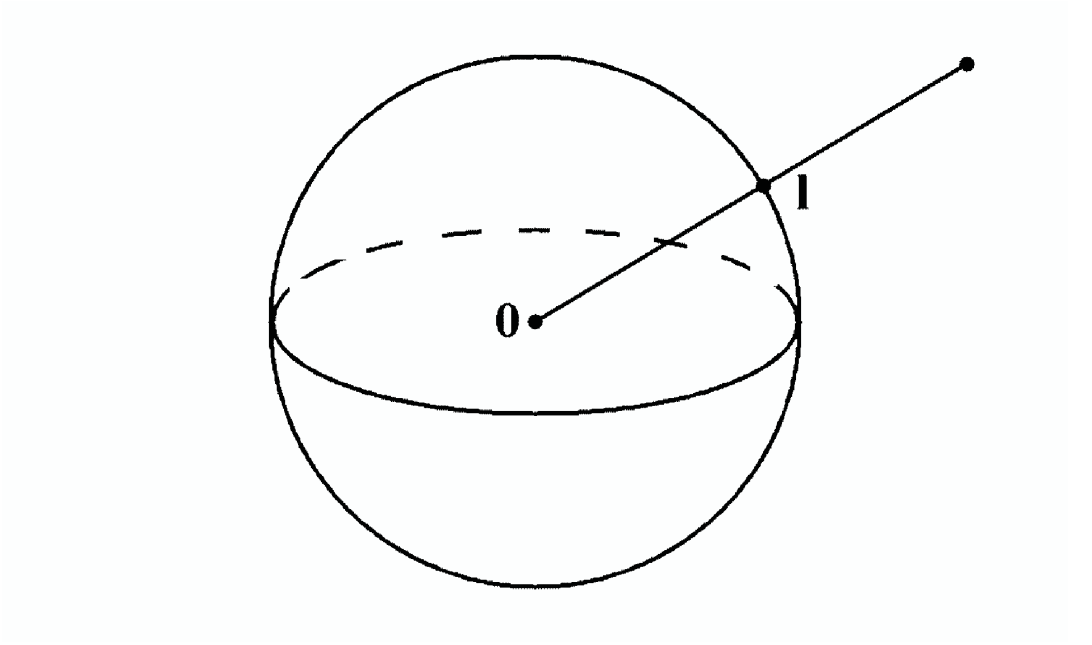


Рис. 9. Множество примеров в $2n$-мерном пространстве

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 ε-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 ε-окрестности вновь "ищем" более сложную точку x_2 и т.д. Останавливали наш процесс поиска, когда в ε-окрестности не удавалось найти "более сложную точку". Таким образом, была выделена совокупность "сложных" примеров относительно **алгоритма А**.

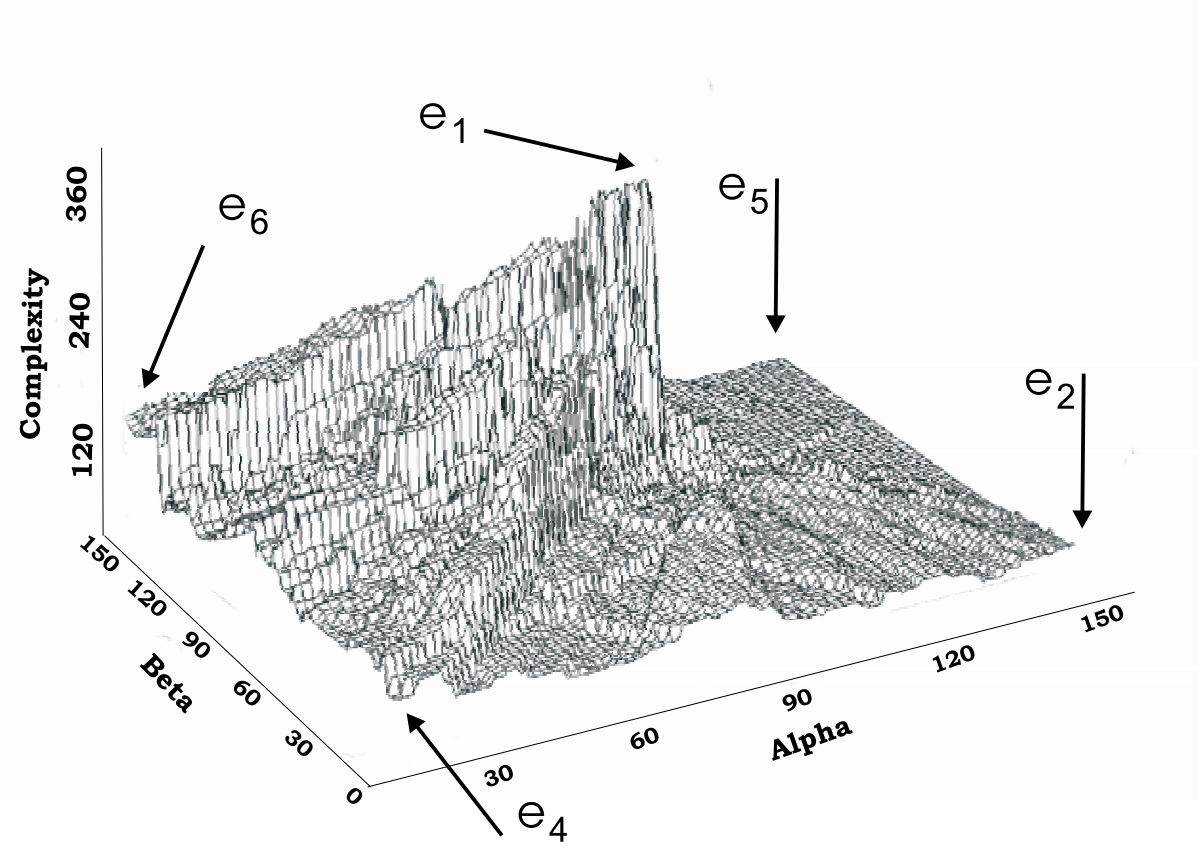


Рис. 10. Окрестность “сложной точки”

На рис. 10 показана окрестность “около сложной точки”. Замечен интересный факт: “сложные” точки “почти” ортогональны между собой.

Глава 11.
Аппроксимационная схема решения задач теории расписаний

В данной главе 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 задачи $\langle\{p_j, d_j\}_{j\in N}, 0\rangle$ с параметрами требований $p_j,\ d_j,\ j \in N$. Пусть π^* – 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 λ. Например, $p_j(\lambda) = \lambda p_j$ или $d_j(\lambda) = \lambda d_j,\ \forall j$. Тогда целевая функция будет является функцией от одного параметра λ, а $\pi^*(\lambda)$ – оптимальные расписания для соответствующего λ, т.е. $F(\lambda) = F(\pi^*(\lambda))$.

Если мы не можем решить исходный пример, то есть 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 приближенной значение целевой функции исходного примера с помощью интерполирования. Пусть модифицированный пример $< \{\lambda p_j, d_j\}_{j\in N}, 0 >$, $\lambda \in (0, +\infty)$. Решением исходного примера будет $F(1)$.

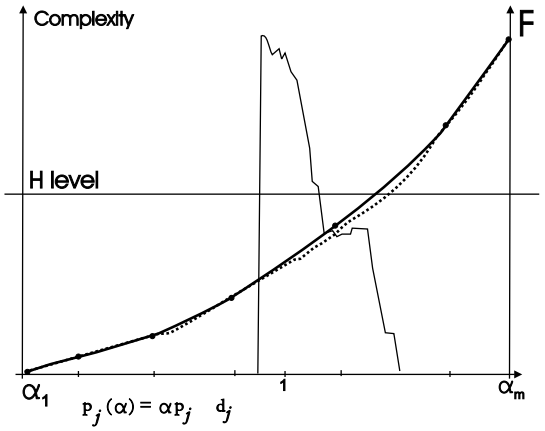


Рис. 11. Аппроксимационная схема решения задачи

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

В **алгоритме B-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 трудоёмкость решения примеров существенно меньше, $\lambda_1, \ldots, \lambda_m \in [\lambda_a, \lambda_b]$. На рис. 11 показан график изменения сложности примеров (количество вершин в дереве поиска алгоритма A) и изменение значения целевой функции относительно изменения параметра λ. Интерполяционный полином Лагранжа строится по точкам $(\lambda_1, F(\lambda_1)), \ldots, (\lambda_m, F(\lambda_m))$. Во всех точках интерполирования $H(\lambda_i) \le \bar{H},\ i = 1, \ldots, m$, т.е. сложность примеров не превышает "порогового" значения трудоёмкости $\bar{H}$.

Заключение.

Для частного случая задачи предложены псевдополиномиальные и полиномиальные алгоритмы. Для NP-трудного случая **B-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 трудоёмкость. Вызывает сомнение, что такими алгоритмами можно решать примеры размерностью $n \geq 100$.

Предложен новый **гибридный алгоритм**, который, по результатам экспериментов, существенно эффективнее **алгоритма ACO** для тестовых примеров Поттса и Ван Васенхова [9].

Для NP-трудных канонических DL-примеров "хорошая эффективность" **гибридного алгоритма** и **ACO** достигается за счет локального поиска.

При оценке эффективности алгоритмов для задачи $1||\sum T_j$ не достаточно использовать тестовые примеры из работы [9].

Библиография.

[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., Струсевич 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., 1969. № 17. P. 701–715.