© 2020 г. Е.Р. ГАФАРОВ, д-р физ.-мат. наук (axel73@mail.ru)
(Институт проблем управления им. В.А. Трапезникова РАН, Москва),
А.А. ЛАЗАРЕВ, д-р физ.-мат. наук (jobmath@mail.ru)
(Институт проблем управления им. В.А. Трапезникова РАН, Москва),
Ф. ВЕРНЕР, профессор (frank.werner@mathematik.uni-magdeburg.de)
(Университет Отто-фон-Герике, Магдебург, Германия)

МИНИМИЗАЦИЯ СУММАРНОГО ВЗВЕШЕННОГО ЗАПАЗДЫВАНИЯ НА ОДНОМ ПРИБОРЕ С РАВНЫМИ ПРОДОЛЖИТЕЛЬНОСТЯМИ ОБСЛУЖИВАНИЯ ТРЕБОВАНИЙ[1]

Рассматривается задача теории расписаний, в которой необходимо минимизировать суммарное взвешенное запаздывание на одном приборе с равными продолжительностями обслуживания требований и неодновременным поступлением требований на обслуживание. Эта задача упомянута как минимальная, статус вычислительной сложности которой неизвестен: http://www2.informatik.uni-osnabrueck.de/knust/class/dateien/classes/ein_ma/ein_ma. Последние результаты по данной задаче опубликованы в 2000 и 2005 гг., а именно, алгоритмы решения частных случаев задачи. В данной статье представлены некоторые свойства задачи и пути дальнейших исследований.

Ключевые слова: теория расписаний, один прибор, суммарное взвешенное запаздывание.

DOI: 10.31857/S0005231020050086

1. Введение

Формулировка задачи: дано множество $N = \{1, \ldots, n\}$ из n требований, которые должны быть обслужены на одном приборе. Прерывание обслуживания требований не допускается. Одновременно прибор может обслуживать не более одного требования. Прибор готов к обслуживанию требований с момента времени 0. Для каждого требования $j \in N$ заданы фиксированное время обслуживания $p_j = p \in Z^+$, директивный срок $d_j \in Z^+$, время поступления $r_j \in Z^+$ (т.е. наиболее раннее время начала обслуживания) и вес $w_j \in Z^+$.

Будем называть активным такое расписание, при котором ни одно требование не может быть перемещено на более ранний срок обслуживания без нарушения ограничений задачи. В данной статье рассматриваются только активные расписания.

Расписание π однозначно определяется перестановкой требований из множества N. Пусть $C_j(\pi)$ – время завершения обслуживания требования j при расписании π. Если $C_j(\pi) > d_j$, тогда требование j запаздывает и принимается $U_j = 1$, иначе $U_j = 0$. Если $C_j(\pi) \leqslant d_j$, тогда требование j не запаздывает.

[1] Исследование выполнено при частичной поддержке гранта Российского научного фонда (проект № 17-19-01665).

Пусть $T_j(\pi) = \max\{0, C_j(\pi) - d_j\}$ — запаздывание требования j при расписании π. Определим $S_j = C_j - p$ как время начала обслуживания требования j при расписании π.

В задаче минимизации суммарного взвешенного запаздывания на одном приборе с равными продолжительностями обслуживания требований и неодновременным поступлением требований на обслуживание необходимо построить оптимальное расписание π^*, при котором достигается минимум функции

$$\sum_{j=1}^{n} f_j(C_j) = \sum_{j=1}^{n} w_j T_j.$$

Обозначим данную задачу как $1|r_j, p_j = p|\sum w_j T_j$ в соответствии с обозначением $\alpha|\beta|\gamma$ задач теории расписаний, представленным в работе [1], где α описывает характеристики приборов, β описывает характеристики требований и ограничения, а γ — целевую функцию. Если $w_j = 1$, $j \in N$, то этот частный случай обозначим как $1|r_j,\ p_j = p|\sum T_j$. В задаче минимизации взвешенного числа запаздывающих требований необходимо минимизировать значение функции

$$\sum_{j=1}^{n} f_j(C_j) = \sum_{j=1}^{n} w_j U_j.$$

Обозначим эту задачу как $1|r_j, p_j = p|\sum w_j U_j$.

В [2, 3] предложены полиномиальные алгоритмы решения задач $1|r_j, p_j = p|\sum T_j$ и $1|r_j, p_j = p|\sum w_j U_j$. Оба алгоритма основаны на следующих свойствах:

- известно множество времен начала обслуживания требований

$$\Theta = \{t : t = r_j + kp, j \in N, k = 0, 1, \ldots, n-1\}.$$

  Мощность множества $|\Theta| \leqslant n^2$;
- известны отношения доминирования между требованиями. Существует оптимальное расписание, при котором требование j обслуживается после всех требований i, где $r_i \leqslant S_j$ и $d_i \leqslant d_j$.

Частный случай, задача $1|r_j, p_j = p|\sum w_j T_j$, с общим директивным сроком может быть решен с помощью алгоритмов [3]. Для данной задачи существует оптимальное расписание, при котором требование j обслуживается после всех требований i, где $r_i \leqslant S_j$ и $w_i \geqslant w_j$.

Частный случай задачи $1|r_j, p_j = p|\sum w_j T_j$, при котором все значения r_j кратны p, может быть сведен к задаче о назначениях и решен за полиномиальное время.

В [4] авторы рассматривают одноприборную задачу теории расписаний — минимизацию суммарного взвешенного штрафа за отклонение от директивного срока $1|p_j = p|\sum(\alpha_j E_j + \beta_j T_j)$, где $E_j(\pi) = \max\{0, d_j - C_j(\pi)\}$. Авторы представили постановку задачи как задачи целочисленного линейного программирования (ЦЛП), предложили схему ее релаксации, для которой построен полиномиальный алгоритм решения. В задаче ЦЛП определены переменные $x_{j,t}$ для каждого требования j и возможного времени завершения

обслуживания требования t, где $x_{j,t} = 1$, если обслуживание требования j заканчивается в момент времени t, и $x_{j,t} = 0$ иначе. В релаксированной задаче $x_{j,t} \in [0, 1]$. Авторы представили полиномиальный алгоритм преобразования решения релаксированной задачи в целочисленное решение исходной задачи с тем же значением целевой функции.

В [5] авторы представили аналогичную модель ЦЛП и аналогичную релаксацию для задачи $1|r_j, p_j = p|\sum w_j T_j$. Авторы утверждают, что если в задаче нет двух требований i и j, для которых $d_i < d_j$ и $w_i < w_j$, то построенный полиномиальный алгоритм преобразует решение релаксированной задачи в оптимальное целочисленное решение исходной задачи. К сожалению, в работе не представлен подобный алгоритм, а дана ссылка на алгоритм из [4], хотя в отличие от задачи $1|r_j, p_j = p|\sum w_j T_j$ в задаче $1|p_j = p|\sum(\alpha_j E_j + \beta_j T_j)$ требования поступают на обслуживание одновременно, а множество времен окончания обслуживания требований относятся к другому интервалу.

Обзор результатов по задачам для параллельных машин с одинаковыми продолжительностями обслуживания требований представлен в [6].

В разделе 2 представлены найденные свойства задачи. В разделе 3 рассмотрены подходы к ее решению. В разделе 4 представлены алгоритмы решения частных случаев, а в разделе 5 — результаты исследования вычислительной сложности. Некоторые задачи максимизации рассмотрены в разделе 6.

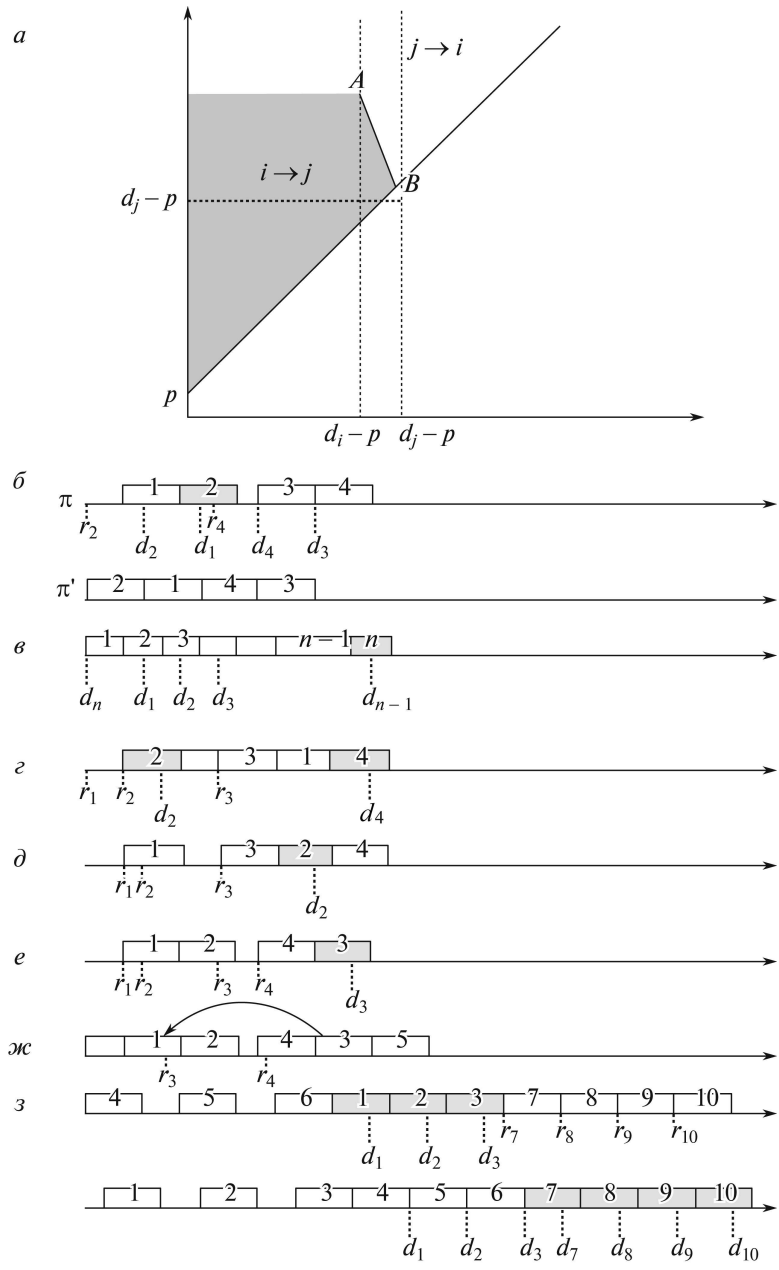
2. Свойства задачи $1|r_j, p_j = p|\sum w_j T_j$

Рассмотрим отношения предшествования между двумя требованиями i и j в оптимальном расписании.

Свойство 1. Пусть t_1 — наиболее ранний момент времени из моментов начала обслуживания требований $\{i, j\}$ и t_2 — поздний момент времени, где $t_2 \geqslant t_1 + p$. Тогда выполняются следующие отношения предшествования.

1. *Если $w_i \geqslant w_j$, $d_i \leqslant d_j$ и $t_1 \geqslant r_i$, тогда требование i обслуживается раньше требования j.*
2. *Если $w_i < w_j$, $d_i \leqslant d_j$ и $t_1 \geqslant \max\{r_i, r_j\}$, то*
   (а) *если $t_2 + p \leqslant d_j$ (требование j не запаздывает, если оно обслуживается начиная со времени t_2), тогда требование i обслуживается раньше требования j;*
   (б) *если $t_1 > d_j - p$ (оба требования запаздывают), тогда требование j обслуживается раньше требования i;*
   (в) *пусть $d_i - p < t_1 \leqslant d_j - p$ и пусть при расписании $\pi = (\pi_1, i, \pi_2, j, \pi_3)$ имеем $S_i(\pi) = t_1$ и $S_j(\pi) = t_2$, а при расписании $\pi' = (\pi_1, j, \pi_2, i, \pi_3)$ имеем $S_i(\pi') = t_2$ и $S_j(\pi') = t_1$. Тогда требование i обслуживается раньше требования j, если выполняется*

$$\sum w_j T_j(\pi) < \sum w_j T_j(\pi') \Longleftrightarrow w_i(t_2 - t_1) - w_j(t_2 + p - d_j) > 0$$
$$\Longleftrightarrow t_2 < \frac{w_j(d_j - p) - w_i t_1}{w_j - w_i};$$



Примеры.

(г) *пусть* $t_1 + p \leqslant d_i \leqslant d_j < t_2 + p$. *Тогда требование* i *обслуживается раньше требования* j, *если выполняется*

$$\sum w_j T_j(\pi) < \sum w_j T_j(\pi') \iff w_i(t_2 + p - d_i) > w_j(t_2 + p - d_j)$$
$$\iff t_2 < \frac{w_j(d_j - p) - w_i(d_i - p)}{w_j - w_i}.$$

Эти отношения предшествования изображены на рисунке, а, где начало координат — точка $(r_{\max}, r_{\max})$, $r_{\max} = \max\{r_i, r_j\}$, *точка* A *есть*

$$A = \left(d_i - p; \frac{w_j(d_j - p) - w_i(d_i - p)}{w_j - w_i} \right)$$

и точка B *есть точка пересечения двух линий*

$$t_2 = t_1 + p \qquad \text{и} \qquad t_2 = \frac{w_j(d_j - p) - w_i t_1}{w_j - w_i}.$$

Точки ниже линии $t_2 = t_1 + p$ *не рассматриваются.*

Отметим, что если $t_1 < r_j$, то сложно установить подобные отношения предшествования, так как при расписании π' требования, которые обслуживаются в интервале времени $[t_1 + p, t_2)$, могут быть смещены в обслуживании на более поздний срок.

В алгоритме динамического программирования, представленного в [3], рассматриваются интервалы $[s, e]$, $s, e \in \Theta$, в которых обслуживаются требования подмножества $N' = \{1, \ldots, k\}$, $k \leqslant n$, т.е. не более k требований в каждом из интервалов. Трудоемкость алгоритма динамического программирования полиномиально зависит от количества таких интервалов и равна $O(n^4)$ операций, так как $|\Theta| = O(n^2)$. Возникает вопрос, как много точек e необходимо рассмотреть для фиксированного значения s, т.е. существует ли пример, в котором нужно рассмотреть порядка $O(n^2)$ точек e?

Свойство 2. Для фиксированных s *и* k *существует порядка* $O(k^2)$ *точек* $e \in \Theta$.

Доказательство. Существует пример, в котором

$$\frac{k}{4} = \left\lceil \frac{k}{4} \right\rceil \qquad \text{и} \qquad p \gg k.$$

Для первых $\frac{k}{4}$ требований пусть

$$r_1 = s, \quad r_2 = r_1 + 2p, \quad r_3 = r_2 + 2p, \quad \ldots,$$

т.е.

$$r_i = s + 2p(i-1), \quad i = 1, \ldots, \frac{k}{4}.$$

Для следующих $\frac{k}{4}$ требований пусть

$$r_i = r_{i-\frac{k}{4}+1} + 1, \quad i = \frac{k}{4}, \frac{k}{4} + 1, \ldots, \frac{k}{2}.$$

Для следующих $\frac{k}{2}$ требований пусть

$$r_i = \frac{k}{2}p + 1 + \left(i - \frac{k}{2} \right), \quad i = \frac{k}{2} + 1, \ldots, k.$$

При активном расписании от $\frac{k}{4}$ до $\frac{k}{2}$ активных требований обслуживаются в начале расписания перед временем $r_{\frac{k}{2}+1}$, а от $\frac{k}{2}$ до $\frac{3k}{4}$ требований обслуживаются начиная с момента времени r_i, $i \in [\frac{k}{2}+1, k]$, без простоя прибора.

Тогда для данного примера существует $O(k^2)$ активных расписаний и $O(k^2)$ возможных моментов времени окончания обслуживания e.

Свойство 3. Пусть

$$T = \left\{ t : t \in \Theta \bigcap [t_1, t_1 + p] \right\},$$

т.е. T есть подмножество точек, входящих в интервал длины p. Тогда $|T| = O(n)$.

3. Алгоритмы решения задачи

Рассматриваемая задача может быть решена с помощью алгоритма динамического программирования и функциональных равенств, представленных в [7].

Пусть дано расписание, при котором первые k обслуживаемых требований составляют подмножество $N \setminus J$, а оставшиеся $n-k$ требований составляют подмножество J. Пусть функция $F(J,t)$ задает минимальное суммарное взвешенное запаздывание требований из подмножества J при условии, что обслуживание этих требований начинается не ранее момента времени $t \in \Theta$. В алгоритме динамического программирования необходимо вычислить значение $F(N,0)$ и найти соответствующее расписание. Функциональные равенства, предложенные в [7], имеют следующий вид:

$$F(J,t) = \min_{i \in J} \left\{ w_i \max\{0, t+p-d_i\} + F(J \setminus \{i\}, t+p) \right\}$$

и $F(\emptyset, t) = 0, \forall t \in \Theta$.

В алгоритме динамического программирования вычисления производятся согласно данным функциональным равенствам. Трудоемкость алгоритма — $O(n^3 2^n)$ операций.

Альтернативный алгоритм решения имеет такую же вычислительную сложность. В этом алгоритме рассматриваются 2^n подмножеств $N' \subseteq N$, где

$$|r_{j_1} - r_{j_2}| \geqslant p, \quad j_1, j_2 \in N'.$$

Для каждого подмножества N' примем $S_j = r_j$, $j \in N'$, и для каждого требования $i \in N \setminus N'$ возможные моменты времени начала обслуживания требований принадлежат множеству Θ', являющемуся подмножеством из $n - |N'|$ наименьших значений множества

$$\left\{ t : t = r_j + kp, j \in N', k = 1, \ldots, n - |N'| \right\}.$$

Тогда данная задача может быть сведена к задаче о назначениях, которая может быть решена за время $O((n-|N'|)^3)$.

Алгоритм, представленный в [3] для частного случая с равными весами требований, может быть использован как эвристика для решения исходной задачи. Несложно представить пример исходной задачи, для которого алгоритм из [3] не вернет оптимальное решение. Пусть в данном алгоритме зафиксировано отношение предшествования, где обслуживание требования j предшествует обслуживанию всех требований i, таких, что $r_i \leqslant S_j$ и $w_i \geqslant w_j$. Тогда несложно представить пример, где при оптимальном расписании требование j предшествует требованию i и

$$C_j - d_j = 1, \qquad C_i - d_i = 1, \qquad C_j < C_i.$$

3.1. Алгоритмы локального поиска

Локальный поиск — это эвристический метод решения задач комбинаторной оптимизации. В этом методе из исходного решения с помощью локальных модификаций строится альтернативное допустимое решение. Эта модификация повторяется до тех пор, пока не найден локальный оптимум или достигнут лимит выполненных операций.

В данной работе рассмотрим следующие модификации расписания. Обозначим активное расписание следующей последовательностью обслуживания требований $\pi = (\pi_1, i, \pi_2, j, \pi_3)$:

- **левый сдвиг**. Обслужить требование j сразу после частичного выполнения частичного расписания π_1, т.е. имеем модифицированное расписание $\pi' = (\pi_1, j, i, \pi_2, \pi_3)$;
- **правый сдвиг**. Обслужить требование i сразу после частичного выполнения частичного расписания π_2, т.е. имеем модифицированное расписание $\pi' = (\pi_1, \pi_2, i, j, \pi_3)$;
- **парная перестановка**. Поменять местами требования i и j в последовательности обслуживания требований, т.е. $\pi' = (\pi_1, j, \pi_2, i, \pi_3)$.

Возникает вопрос: “Можно ли с помощью данных модификаций найти оптимальное расписание, причем при каждой модификации значение целевой функции не должно увеличиваться.”

Свойство 4. Существуют пример задачи и расписание, при котором каждый левый сдвиг увеличивает значение целевой функции, но применение нескольких левых сдвигов уменьшает значение целевой функции.

Доказательство. Рассмотрим пример, где

$$r_1 = 2,\ r_2 = 0,\ r_3 = 9,\ r_4 = 6,\ p = 3,\ d_1 = 6,\ d_2 = 3,\ d_3 = 12,\ d_4 = 9,$$
$$w_1 = w_3 = 100,\ w_2 = w_4 = 1$$

(см. рисунок,b). Пусть исходное расписание $\pi = (1, 2, 3, 4)$. При левом сдвиге требования 2 на позицию перед требованием 1 значение целевой функции увеличивается. Аналогично при левом сдвиге требования 4 на позицию перед требованием 3 значение целевой функции увеличивается. Но при применении обоих левых сдвигов будет получено расписание $\pi' = (2, 1, 4, 3)$, при котором значение целевой функции уменьшается. При расписании π' суммарное взвешенное запаздывание равно 0, а при расписании π имеем суммарное взвешенное запаздывание, равное 11.

Следствие 1. Пусть π — расписание, при котором каждый левый сдвиг увеличивает значение целевой функции. Тогда относительная погрешность расписания π может быть сколь угодно большой.

Аналогичное доказательство можно представить для следующего свойства.

Свойство 5. Существуют пример задачи и расписание, при котором каждый правый сдвиг увеличивает значение целевой функции, но применение нескольких правых сдвигов уменьшает значение целевой функции.

Свойство 6. Пусть π — некоторое расписание и $N' = \{j_1, \ldots, j_k\}$ — подмножество требований, которые должны быть сдвинуты влево, и пусть порядок обслуживания требований из N' фиксирован, т.е. обслуживание требования j_1 предшествует обслуживанию требования j_2, требование j_2 предшествует требованию j_3 и т.д. Тогда оптимальная модификация расписания путем левого сдвига каждого требования из N' может быть найдена за $O(n^8)$ операций.

Доказательство. Оптимальная модификация может быть найдена с помощью следующего алгоритма динамического программирования. На каждой стадии $l = k,\ k-1, \ldots, 1$ для каждого требования $j_l \in N'$ необходимо рассмотреть x_l позиций в перестановке, в которой порядок обслуживания требований из $N \setminus N'$ остается неизменным и все возможные моменты начала обслуживания $t_l \in \Theta$. Таким образом, состояние системы в алгоритме динамического программирования определяется вектором (x_l, t_l). На каждой стадии необходимо вычислить функцию $F_l(x_l, t_l)$, являющуюся суммарным взвешенным запаздыванием требований, обслуживаемых при активном расписании с момента t_l согласно последовательности, начинающейся с требования j_l. Эта функция используется на следующей стадии для требования j_{l-1}. На каждой стадии необходимо рассмотреть $O(n^{1+2})$ состояний системы. Для каждого состояния системы (x_l, t_l) необходимо рассмотреть состояния системы, полученные на предыдущей стадии, чтобы за $O(n)$ операций вычислить значение функции $F_l(x_l, t_l)$. Таким образом, трудоемкость алгоритма динамического программирования составляет $O(n^8)$ операций.

Свойство 7. Пусть для расписания $(1, 2, 3)$ две попарные перестановки $1 \Longleftrightarrow 2$ и $2 \Longleftrightarrow 3$ приводят к увеличению значения целевой функции. Тогда существует пример, для которого попарная перестановка $1 \Longleftrightarrow 3$ приводит к уменьшению значения целевой функции.

Параметры данного примера:

$$r_1 = r_2 = r_3 = 0,\ p = 3,\ d_1 = 5,\ d_2 = 7,\ d_3 = 8,\ w_1 = 1,\ w_2 = w_3 = 5.$$

При расписании $(1, 2, 3)$ имеем $\sum w_j T_j = 5$, а при расписании $(3, 2, 1)$ имеем $\sum w_j T_j = 4$.

Следствие 2. При локальном поиске с помощью попарной перестановки необходимо рассмотреть $O(n^2)$ пар требований.

Свойство 8. Пусть для расписания $(1, \ldots, n-1, n)$ любая попарная перестановка приводит к увеличению значения целевой функции. То-

гда существует пример, в котором левый сдвиг приводит к расписанию $(n, 1, \ldots, n-1)$ *с меньшим значением целевой функции.*

Доказательство. См. рисунок, *в*. Рассмотрим следующий пример:

$$(1)\qquad \begin{cases} n > 3, & \\ r_i = 0, & i = 1\ldots, n, \\ p = 2, & \\ w_i = pw_n - 1, & i = 1\ldots, n-2, \\ w_{n-1} = pw_n + 1, & i = 1\ldots, n-2, \\ d_n = 0, & \\ d_i = (i+1)p - 1, & i = 1\ldots, n-1. \end{cases}$$

При расписании $\pi = (1, \ldots, n-1, n)$ любая попарная перестановка приводит к увеличению значения целевой функции. Но при расписании $\pi' = (n, 1, \ldots$ $\ldots, n-1)$ имеем

$$F(\pi') = F(\pi) + \sum_{i=1}^{n-1} w_n - p(n-1)w_n = \\ = F(\pi) + p(n-1)w_n - (n-2) + 1 - p(n-1)w_n < F(\pi).$$

Свойство 9. Существуют пример и расписание, при котором любой левый сдвиг приводит к увеличению значения целевой функции, но возможен правый сдвиг, который приводит к уменьшению значения целевой функции.

3.2. Релаксация задачи

Рассмотрим релаксацию задачи, при которой все значения r_i округлены таким образом, что расстояние между ними кратно p. Например,

$$r'_j = \left\lfloor \frac{r_j}{p} \right\rfloor p, \qquad d'_j = d_j - (r_j - r'_j).$$

Такой модифицированный пример I' может быть сведен к задаче о назначениях и решен за полиномиальное время. Пусть π — оптимальное расписание для исходного примера I и π' — оптимальное расписание для модифицированного примера I' с округленными параметрами.

Рассмотрим вопросы:

- какова относительная погрешность

$$\frac{F(\pi) - F(\pi')}{F(\pi)}$$

  для примера I?
- какова разница значений $F(\pi)$ для примера I и $F'(\pi')$ для примера I', где F' — значение целевой функции для примера с округленными параметрами?

Свойство 10. Существует пример, для которого значение

$$\frac{F(\pi)}{F'(\pi')}$$

может быть сколь угодно большим.

Рассмотрим следующий пример, где $n = 2$, $p = 3$, $r_1 = 1$, $r_2 = 3$, $d_1 = 4$, $d_2 = 6$, $w_1 = w_2 = 1$. При расписании $\pi = \pi' = (1, 2)$ имеем $F(\pi) = 1$, а также $F'(\pi') = 0$, где $r'_1 = 0$, $d'_1 = 3$, $r'_2 = r_2$, $d'_2 = d_2$.

Свойство 11. Существует пример, для которого

$$\frac{F(\pi) - F(\pi')}{F(\pi)}$$

может быть сколь угодно большим.

Рассмотрим следующий пример:

$$(2) \qquad \begin{cases} p = 4, \\ r_n = 0, \\ r_i = \dfrac{p}{2}, & i = 1 \ldots, n-1, \\ d_i = r_i + 2p - 1, & i = 1 \ldots, n-1, \\ d_n = p, \\ w_n = 1, \\ w_i = np, & i = 1 \ldots, n-1. \end{cases}$$

Расписание $\pi' = (1, \ldots, n-1, n)$ является оптимальным для примера I', при котором запаздывает только требование n. Для примера I при этом расписании также запаздывает только требование n. При расписании $\pi = (n, 1, \ldots, n-1)$, оптимальном для примера I, ни одно из требований не запаздывает.

Можно рассмотреть другую релаксацию, при которой значения r'_i вычисляются таким образом, чтобы минимизировать значение $\sum_{j=1}^{n} |r_j - r'_j|$. Для такой релаксации свойства 10 и 11 также верны. Чтобы доказать это, необходимо рассмотреть n дополнительных требований, для которых $r_j = np$, $j = n+1, \ldots, 2n$.

4. Частные случаи задачи $1|r_j, p_j = p| \sum w_j T_j$

Представим алгоритмы решения двух частных случаев задачи. Как было сказано выше, частный случай задачи с общим директивным сроком может быть решен за полиномиальное время.

Рассмотрим частный случай, в котором разница максимального и минимального директивных сроков составляет $d_{\max} - d_{\min} \leqslant p$. Этот частный случай может быть решен с помощью следующего алгоритма.

Выберем два пограничных требования j_1 и j_2, которые будут обслужены при активном расписании одно за другим начиная с момента времени S_{j_1}

так, что никакое другое требование не может быть обслужено в интервале $[d_{\min}, d_{\max}]$, т.е.

$$S_{j_1} \leqslant d_{\min}, \quad C_{j_2} \geqslant d_{\max}.$$

Тогда подзадача со множеством требований $N \setminus \{j_1, j_2\}$ может быть решена модификацией алгоритма, представленного в [3], за $O(n^7)$ операций, где требование j обслуживается после обслуживания всех требований i, для которых $r_i \leqslant S_j$ и $w_i \geqslant w_j$. В модифицированном алгоритме рассматриваются только интервалы $[s, e]$, которые не пересекаются с интервалом $[S_{j_1}, C_{j_2}]$.

Имеется $O(n^2)$ пар пограничных требований j_1 и j_2, $O(n)$ возможных моментов начала обслуживания $S_{j_1} \in \Theta$ согласно свойству 3 и единственное значение S_{j_2} для выбранного S_{j_1}, так как рассматриваются только активные расписания. Тогда данный частный случай может быть решен за $O(n^{2+1+7})$ операций.

Далее рассмотрим частный случай с n требованиями, где

$$\begin{cases} w_1 \leqslant w_2 \leqslant \ldots \leqslant w_{n-1}, \\ d_1 \geqslant d_2 \geqslant \ldots \geqslant d_{n-1}. \end{cases} \tag{3}$$

Для решения данного частного случая необходимо выбрать время начала обслуживания $S_n \in \Theta$ и затем решить оставшуюся подзадачу с помощью модифицированного алгоритма из [3]. То есть данный частный случай может быть решен за $O(n^{2+7})$ операций.

5. Результаты численных экспериментов

В разделе представлены результаты численных экспериментов, проведенных с целью выяснить количество возможных точек в Θ, если для k требований время начала обслуживания фиксировано.

При этом был рассмотрен следующий частный случай задачи:

$$\begin{cases} w_1 \leqslant w_2 \leqslant \ldots \leqslant w_n, \\ d_1 \leqslant d_2 \leqslant \ldots \leqslant d_n \end{cases} \tag{4}$$

с $n = 10$.

Числовые параметры примеров были сгенерированы следующим образом. Для каждого значения $p \in \{5, 10, 15, 20, 25, 30\}$ были рассмотрены 10 примеров, для которых значения $r_j \in [0, (n-2)p]$, $d \in [0, (n-1)p]$, $w \in [1, 120]$ были определены случайным образом согласно равномерному распределению. Для каждого примера был использован алгоритм ветвей и границ с подпрограммой ветвления (j, Π), где j — требование, для которого в подпрограмме необходимо определить время начала обслуживания, и Π — частично построенное расписание, при котором моменты начала обслуживания S_i, $i = 1, \ldots, j-1$, уже определены. В данной подпрограмме вычисляется множество Θ_j согласно Π и значению r_j. Пусть $S_{j'} < S_{j''}$, $j', j'' \leqslant j-1$, и нет других требований

$j''' \leqslant j-1$, для которых $S_{j'} < S_{j'''} < S_{j''}$. Тогда определим множества:

$$\Omega_1 = \{S_{j'} + kp, k = 1, \ldots, n-j : S_{j'} + kp + p \leqslant S_{j''}, S_{j'} + kp \geqslant r_j\},$$
$$\Omega_2 = \{r_i + kp, k = 1, \ldots, n-j, i \geqslant j, r_j \leqslant r_i + kp \leqslant S_{j''} - p, r_i > S_{j'} + p\} \text{ и}$$
$$\Omega_3 = \{S_{j''} - kp, k = 1, \ldots, n-j : S_{j''} - kp - p \geqslant S_{j'}, S_{j''} - kp \geqslant r_j\}.$$

Пусть t — максимальное значение из множества Ω_2, где $(S_{j''} - t)$ кратно значению p. Если такое t существует, тогда из множества Ω_2 удаляются все значения, которые больше t, и из множества Ω_3 удаляются все значения, которые меньше t. Тогда множество Θ_j состоит из элементов множеств $\Omega_1, \Omega_2, \Omega_3$ и значения r_j, если $S_{j'} + p < r_j < S_{j''} - p$.

В подпрограмме ветвления рассматриваются все возможные моменты времени $t \in \Theta_j$. Пусть $TWT(\Pi)$ — суммарное взвешенное запаздывание требований, обслуживаемых при частичном расписании Π. Если

$$TWT(\Pi) + w_j \max\{t + p - d_j, 0\} \geqslant UB,$$

тогда t исключается из рассмотрения, где верхняя оценка UB — лучшее (минимальное) найденное на данный момент значение целевой функции.

Представим результат численного эксперимента для одного примера.

$$p = 10, r_j \in \{72, 58, 29, 56, 49, 58, 55, 70, 57, 72\},$$
$$d_j \in \{66, 71, 73, 75, 76, 76, 82, 85, 88, 88\},$$
$$w_j = \{22, 30, 42, 56, 68, 75, 75, 94, 103, 111\}.$$

Для данного примера в алгоритме ветвей и границ общее количество ветвей к рассмотрению 33 346 541, на каждом шаге (уровне дерева поиска) $j = 1, \ldots, n$ рассмотрено

$$\{58; 1\,926; 43\,059; 588\,149; 3\,820\,730; 9\,432\,283; 14\,004\,668; 4\,636\,929; 818\,539; 200\}$$

точек. При использовании свойства 1 количество ветвей можно сократить до 5 303 348, и на каждом шаге $j = 1, \ldots, n$ рассмотрено

$$\{58; 1\,666; 26\,343; 233\,762; 962\,857; 1\,457\,811; 1\,979\,532; 531\,006; 110\ 311; 2\}$$

точек.

В соответствии с результатами численных экспериментов свойство 1 позволяет сократить количество точек в Θ_j на 47%. В таблице представлены результаты численного эксперимента для примеров, где $p \in \{5, 10, 15, 20, 25, 30\}$. В колонках 1–3, представлены времена поступления, директивные сроки и веса для 10 требований. В колонке 4 представлено расписание (время начала обслуживания требований). В колонке 5 дано оптимальное значение целевой функции TWT для примера. В колонках 7 и 8 представлено количество ветвей в алгоритме ветвей и границ без использования свойства 1 — $TNN1$ — и с использованием свойства 1 — $TNN2$. В колонке 8 представлен процент рассмотренных ветвей при использовании свойства 1.

Заметим, что трудоемкость алгоритма ветвей и границ экспоненциальна, что не позволяет решать примеры размерности $n = 20$ за 60 минут на ПЭВМ Intel Core 2 Duo CPU P8600 2,4 GHz, 4 GB RAM.

Таблица. Результаты для $p \in \{5, 10, 15, 20, 25, 30\}$

r_j	d_j	w_j	Расписание	TWT	$TNN1$	$TNN2$	%
1	2	3	4	5	6	7	8
$p = 5$							
21, 0, 35, 31, 26, 11, 10, 25, 11, 14	8, 26, 35, 35, 38, 40, 42, 43, 44, 44	6, 13, 20, 20, 58, 59, 84, 90, 110, 116	50, 0, 40, 45, 30, 15, 10, 25, 20, 35	782	199344	107496	54
17, 36, 8, 34, 20, 25, 16, 39, 6, 23	19, 19, 28, 32, 32, 37, 38, 39, 43, 44	14, 21, 23, 46, 50, 60, 67, 89, 103, 107	51, 46, 11, 36, 21, 26, 16, 41, 6, 31	2227	620833	298294	48
16, 26, 10, 5, 25, 29, 11, 19, 31, 14	30, 31, 33, 34, 36, 39, 39, 43, 44, 44	9, 15, 44, 50, 52, 64, 77, 89, 91, 113	50, 45, 10, 5, 25, 30, 15, 20, 40, 35	601	196973	112646	57
35, 35, 37, 2, 22, 9, 16, 8, 35, 21	16, 22, 23, 31, 35, 35, 40, 41, 42, 44	22, 27, 31, 54, 65, 77, 84, 105, 105, 108	50, 45, 40, 2, 23, 13, 18, 8, 35, 28	2296	205273	80800	39
7, 18, 10, 31, 4, 7, 4, 20, 19, 0	11, 20, 30, 30, 33, 34, 35, 37, 41, 44	9, 9, 45, 48, 50, 62, 64, 85, 87, 117	41, 46, 10, 31, 5, 15, 20, 25, 36, 0	882	264585	207667	78
$p = 10$							
7, 65, 56, 9, 17, 39, 77, 16, 64, 79	35, 50, 56, 70, 74, 81, 86, 87, 87, 89	39, 46, 54, 55, 64, 71, 84, 97, 104, 104	7, 97, 57, 17, 27, 47, 77, 37, 67, 87	4132	100035	54247	54
53, 63, 4, 23, 26, 27, 18, 10, 49, 28	36, 36, 42, 55, 55, 68, 75, 81, 85, 88	1, 24, 26, 27, 27, 46, 58, 79, 95, 107	94, 84, 4, 24, 34, 44, 54, 14, 64, 74	1460	592976	416717	70
32, 56, 10, 4, 49, 78, 57, 42, 40, 23	41, 51, 67, 72, 83, 84, 84, 85, 87, 89	11, 28, 29, 34, 41, 48, 56, 61, 77, 85	34, 56, 14, 4, 96, 86, 66, 44, 76, 24	1972	414289	316671	76
24, 76, 5, 36, 21, 57, 70, 36, 59, 52	27, 35, 48, 69, 71, 73, 76, 79, 83, 88	9, 23, 26, 34, 71, 72, 102, 105, 113, 114	107, 97, 5, 36, 21, 57, 87, 46, 67, 77	4608	6476173	3329563	51
65, 2, 8, 3, 29, 34, 37, 33, 47, 57	16, 46, 47, 62, 64, 79, 79, 83, 83, 87	25, 31, 32, 49, 79, 90, 94, 100, 101, 108	92, 2, 12, 22, 32, 42, 52, 62, 72, 82	2690	849744	367675	43
$p = 15$							
97, 105, 22, 98, 42, 32, 114, 47, 79, 86	67, 89, 103, 112, 114, 116, 118, 130, 131, 134	5, 36, 36, 47, 47, 48, 53, 103, 112, 116	158, 143, 22, 98, 52, 37, 128, 67, 82, 113	4386	1709603	1166960	68
93, 31, 60, 55, 28, 90, 51, 29, 106, 85	50, 68, 68, 76, 86, 89, 105, 119, 122, 129	12, 13, 17, 27, 50, 52, 60, 63, 63, 90	166, 151, 136, 58, 28, 90, 73, 43, 106, 121	5719	4417446	1328448	30
54, 37, 9, 104, 93, 64, 35, 26, 20, 61	41, 109, 118, 123, 126, 127, 128, 129, 130, 133	6, 11, 20, 35, 45, 56, 59, 63, 91, 109	144, 129, 9, 114, 99, 69, 39, 54, 24, 84	1303	386234	318040	82

Таблица. (продолжение)

1	2	3	4	5	6	7	8
19, 58, 44, 22, 101, 44, 61, 27, 81, 119	65, 70, 79, 80, 88, 90, 102, 112, 122, 134	2, 18, 29, 43, 50, 56, 67, 81, 95, 110	157, 142, 52, 22, 112, 67, 82, 37, 97, 127	4610	1950750	843607	43
18, 51, 73, 14, 57, 50, 72, 57, 56 119	54, 73, 86, 89, 95, 105, 110, 116, 132, 134	1, 13, 20, 35, 44, 68, 85, 95, 96, 111	18, 155, 140, 33, 65, 50, 80, 95, 110, 125	3307	3689263	1928262	52
$p = 20$							
71, 25, 58, 69, 93, 122, 72, 123, 95, 70	102, 116, 125, 140, 142, 151, 164, 166, 166, 175	3, 6, 8, 10, 43, 51, 67, 87, 93, 113	218, 25, 58, 198, 98, 178, 78, 138, 118, 158	3924	29709961	6809732	23
83, 67, 52, 154, 92, 60, 38, 49, 65, 80	73, 103, 103, 117, 146, 156, 166, 171, 172, 175	18, 18, 47, 58, 67, 69, 77, 81, 105, 118	198, 218, 58, 178, 98, 78, 38, 118, 138, 158	10092	1849948	1203732	65
101, 41, 18, 116, 101, 42, 59, 138, 104, 140	86, 111, 117, 123, 137, 158, 166, 169, 170, 176	2, 19, 23, 24, 25, 27, 52, 59, 110, 110	201, 41, 18, 181, 101, 61, 81, 141, 121, 161	2692	2074218	884291	43
55, 84, 135, 56, 36, 98, 60, 5, 28, 43	83, 98, 106, 106, 111, 112, 130, 149, 165, 170	2, 22, 34, 37, 52, 54, 63, 72, 82, 113	188, 128, 168, 68, 48, 108, 88, 5, 28, 148	5002	758835	471049	62
145, 95, 25, 148, 40, 28, 84, 135, 46, 52	87, 87, 93, 122, 153, 155, 160, 166, 171, 178	18, 29, 36, 38, 42, 54, 75, 84, 90, 96	205, 105, 25, 185, 45, 65, 85, 145, 125, 165	7412	1288063	579338	45
$p = 25$							
5, 93, 118, 175, 140, 129, 22, 63, 20, 74	121, 129, 146, 150, 167, 168, 169, 173, 178, 211	26, 31, 32, 51, 52, 56, 84, 89, 96, 106	5, 105, 230, 205, 155, 130, 30, 80, 55, 180	8275	46808	29428	63
193, 142, 193, 194, 168, 175, 107, 186, 164, 59	152, 197, 200, 200, 218, 219, 220, 222, 222, 224	25, 33, 44, 46, 50, 55, 64, 71, 84, 97	317, 142, 292, 267, 242, 217, 107, 192, 167, 59	17845	2103722	444158	21
176, 177, 167, 25, 48, 113, 65, 80, 66, 170	100, 120, 136, 142, 152, 186, 201, 202, 205, 221	5, 17, 26, 28, 58, 81, 100, 106, 109, 118	250, 225, 175, 25, 50, 125, 75, 100, 150, 200	5221	973995	807576	83
137, 41, 14, 164, 146, 186, 74, 36, 119, 74	83, 92, 127, 132, 167, 174, 196, 208, 217, 220	5, 11, 31, 50, 58, 66, 80, 83, 114, 116	246, 41, 14, 171, 146, 221, 91, 66, 119, 196	9240	513204	278014	54

Таблица. (окончание)

1	2	3	4	5	6	7	8
138, 79, 40, 50, 109, 2, 163, 156, 132, 190	72, 105, 110, 151, 153, 169, 200, 204, 208, 221	4, 48, 53, 57, 59, 60, 60, 74, 80, 118	240, 90, 40, 65, 115, 2, 215, 165, 140, 190	3652	217856	173628	80
$p = 30$							
55, 1, 219, 107, 32, 190, 92, 17, 80, 130	119, 129, 143, 150, 180, 184, 198, 253, 255, 263	7, 9, 36, 40, 53, 84, 87, 102, 112, 112	280, 1, 250, 121, 61, 190, 151, 31, 91, 220	9333	641049	342412	53
46, 239, 103, 164, 226, 50, 209, 83, 117, 51	117, 129, 163, 166, 208, 210, 245, 264, 268, 269	19, 28, 52, 53, 79, 81, 84, 106, 110, 114	46, 316, 106, 166, 286, 76, 226, 136, 196, 256	19060	494327	325578	66
139, 13, 155, 191, 31, 182, 175, 158, 53, 101	173, 191, 192, 208, 214, 221, 227, 241, 246, 257	15, 16, 17, 31, 44, 45, 61, 80, 84, 101	139, 13, 289, 259, 43, 229, 199, 169, 73, 103	6502	620830	377214	61
191, 75, 230, 187, 26, 51, 77, 152, 148, 45	112, 119, 154, 220, 222, 230, 232, 242, 246, 266	9, 13, 26, 26, 58, 68, 88, 99, 106, 110	298, 86, 268, 208, 26, 56, 116, 178, 148, 238	6376	536269	269281	50
79, 132, 172, 203, 126, 81, 179, 178, 183, 113	166, 178, 205, 210, 219, 241, 242, 247, 254, 255	2, 13, 20, 27, 66, 83, 95, 98, 114, 119	358, 328, 298, 268, 143, 81, 208, 178, 238, 113	9216	22248262	7730427	35

6. Вычислительная сложность задачи $1|r_j, p_j = p|\sum w_j T_j$

Насколько известно, при доказательстве NP-трудности задачи теории расписаний с классическими ограничениями p_j, w_j, d_j, r_j, D_j и целевыми функциями $C_{\max}, L_{\max}, \sum w_j C_j, \sum w_j T_j, \sum w_j U_j$ используются следующие два подхода:

- сведение к исходной задаче задачи типа “Разбиения” (Одномерный ранец, 3-Разбиение), если принять, что p_j зависят от чисел b_j из задачи о Разбиении. При таком сведении количество моментов времени возможного окончания обслуживания требований не ограничено значением $O(n^2)$;
- сведение к исходной задаче задачи теории графов (например, задачи о клике), если заданы отношения предшествования между требованиями.

Однако оба эти подхода не приводят к доказательству NP-трудности задачи $1|r_j, p_j = p|\sum w_j T_j$.

В упомянутых выше подходах к доказательству NP-трудности рассматриваются частные случаи исходных задач, где структура оптимального расписания известна, см., например, [8]. Однако для задач с равными продолжительностями обслуживания требований при известной структуре оптималь-

ного расписания просто построить полиномиальной алгоритм решения и использованием динамического программирования.

Таким образом,

- **с одной стороны**, задача $1|r_j, p_j = p|\sum w_j T_j$ не содержит сложностей типа экспоненциального количества моментов времени завершения обслуживания требований или отношений предшествования,
- **с другой стороны**, для данного примера неизвестны правила предшествования, позволяющие построить полиномиальный алгоритм решения.

Предполагаем, что задача $1|r_j, p_j = p|\sum w_j T_j$ является NP-трудной, но чтобы доказать это, необходим новый подход к доказательству NP-трудности.

7. Задачи с обратными критериями оптимизации

Обычно в теории расписаний рассматриваются задачи, где необходимо найти минимум некоторой целевой функции. Например, минимизация времени завершения обслуживания всех требований является популярным критерием оптимизации. Также в классических задачах рассматриваются критерии минимизации суммарного запаздывания, минимизации количества запаздывающих требований. В этом разделе рассмотрим задачи с *обратными* критериями оптимизации, а именно: максимизация времени завершения обслуживания всех требований, максимизация суммы моментов времен завершения обслуживания требований, максимизация количества запаздывающих требований. В этих задачах допустимыми являются только активные расписания, иначе задачи были бы тривиальны.

Исследование задач с *обратными* критериями оптимизации имеет теоретическую значимость, сами задачи имеют практическую интерпретацию [9–11]. Максимальное значение времени завершения обслуживания всех требований может быть использовано, чтобы сократить множество Θ в задаче минимизации.

В задаче максимизации времени завершения обслуживания всех требований необходимо найти активное расписание π^*, при котором максимизировано значение $C_{\max}(\pi) = \max_{j=1}^{n}\{C_j\}$. Обозначим данную задачу как $1|r_j, p_j = p|\max C_{\max}$. Аналогично обозначим другие две рассматриваемые задачи как $1|r_j, p_j = p|\max\sum C_j$ и $1|r_j, p_j = p|\max\sum U_j$ — задачи максимизации суммы моментов времен завершения обслуживания требований и максимизации количества запаздывающих требований.

Представим полиномиальный алгоритм решения задач

$1|r_j, p_j = p|\max C_{\max}$ и $1|r_j, p_j = p|\max\sum C_j$ и некоторые свойства задачи $1|r_j, p_j = p|\max\sum U_j$.

7.1. Алгоритм решения задач $1|r_j, p_j = p|\max C_{\max}$ *и* $1|r_j, p_j = p|\max\sum C_j$

Обозначим работы согласно порядку $r_1 \leqslant r_2 \leqslant \ldots \leqslant r_n$. Без потери общности примем $r_1 = 0$.

Идея алгоритма заключается в следующем. В перестановке, однозначно определяющей расписание обслуживания требований, одна за одной рассматриваются позиции. Для каждой позиции выбирается одно требование j в соответствии с r_j так, чтобы сделать промежуток (простой) между временем завершения обслуживания предыдущего требования и r_j как можно больше, но меньше, чем p. Таким образом, ни одно другое требование не может быть обслужено раньше момента времени r_j.

Алгоритм 1.

1. Пусть $t := 0$ — время завершения обслуживания последнего требования, поставленного в перестановку (в расписание), т.е. требования, время обслуживания которого определено, и $N_u := N$ — множество требований, время обслуживания которых не определено. Примем также $\pi := ()$ — найденная частичная перестановка.
2. Для $l := 1$ до n выполнить
   2.1. Выбрать требование $j := \underset{i \in N_u}{\operatorname{argmax}}\{r_i, t \leqslant r_i < t + p\}$.
   2.2. Если такого требования j нет, тогда
   Выбрать требование j из множества N_u, где $r_i \leqslant t$.
   Если такого требования j нет, тогда
   $t := \min\limits_{i \in N_u}\{r_i, t < r_i\}$. Перейти к шагу 2.1.
   2.2. $\pi := (\pi, j)$. $N_u := N_u \setminus \{j\}$. $t := C_j(\pi)$.
3. π — оптимальное расписание.

Трудоемкость алгоритма 1 — $O(n)$ операций, и $O(n \log n)$ необходимо для упорядочивания работ согласно правилу $r_1 \leqslant r_2 \leqslant \ldots \leqslant r_n$.

Лемма 1. Алгоритм 1 строит оптимальное расписание для задач $1|r_j, p_j = p|\max C_{\max}$ *и* $1|r_j, p_j = p|\max \sum C_j$.

Доказательство. Предположим, что существует оптимальное расписание $\pi^* = (l, \pi_1, j, \pi_2)$, при котором первым обслуживается требование l, хотя в алгоритме 1 строится расписание, в котором первое обслуживаемое требование — l. Если $r_l \leqslant r_j$, тогда при расписании $\pi = (j, \pi_1, l, \pi_2)$ имеем $C_i(\pi) \geqslant C_i(\pi^*)$ для кажого требования $i \in N \setminus \{j\}$ и $C_{\max}(\pi) \geqslant C_{\max}(\pi^*)$.

Если $r_j < r_l$, тогда расписание π^* не является активным, так как требование 1 может быть обслужено первым, начиная с момента времени $S_1 = 0$.

Дальнейшее доказательство может быть выполнено по индукции.

7.2. Свойства задачи $1|r_j, p_j = p|\max \sum U_j$

Известно, что задача $1|r_j|\max \sum U_j$ является NP-трудной [11], но ее частный случай $1|r_j = 0|\max \sum U_j$ может быть решен за $O(n^2)$ операций [9]. Для данного частного случая существует оптимальная последовательность обслуживания требований (G, W), при которой все требования из G обслуживаются вовремя, а все требования из W запаздывают и обслуживаются в порядке неуменьшения директивных сроков.

Представим найденные свойства для задачи $1|r_j, p_j = p|\max \sum U_j$.

Свойство 12. Существует оптимальное расписание, при котором обслуживание запаздывающего требования i предшествует обслуживанию не запаздывающего требования j и $r_j < r_i$.

Подобный пример представлен на рисунке, *г*, где требования i равные 2 и 4 запаздывают, а требования j равные 1 и 3 — не запаздывают.

Свойство 13. Существует оптимальное расписание, при котором обслуживание запаздывающего требования i предшествует обслуживанию запаздывающего требования j и $d_i < d_j$.

Подобный пример представлен на рисунке, *г*, где $i = 2$, $d_2 = r_2 + p - 1$ и $j = 1$, $d_1 = r_2 + p$.

Свойство 14. Существует пример, для которого алгоритм 1 строит неоптимальное расписание.

Подобный пример представлен на рисунке, *д*, при котором только требование 2 запаздывает. А на рисунке, *е* показано, что максимальное время простоя не приводит к оптимальному расписанию, где требование 3 — единственное запаздывающее требование.

Для решения задачи предлагается следующая эвристика: если при расписании π требование j обслуживается вовремя и $S_j(\pi) > r_j$, тогда необходимо переставить это требование на более раннее время обслуживания.

Свойство 15. Если при расписании π требование j обслуживается вовремя, а требование i запаздывает и оба требования обслуживаются не ранее чем с момента времени $t \geqslant \max\{r_i, r_j\}$, тогда существует расписание π', где обслуживание требования j предшествует обслуживанию i и $\sum U_j(\pi') \geqslant \sum U_j(\pi)$.

Другими словами, если два требования обслуживаются не ранее чем с момента времени $t \geqslant \max\{r_i, r_j\}$, то при оптимальном расписании обслуживание незапаздывающего требования предшествует обслуживанию запаздывающего.

Свойство 16. Если при расписании π оба требования j и i запаздывают, $d_j \leqslant d_i$ и оба требования обслуживаются не ранее чем с момента времени $t \geqslant \max\{r_i, r_j\}$, тогда существует расписание π', где обслуживание требования j предшествует обслуживанию i и $\sum U_j(\pi') \geqslant \sum U_j(\pi)$.

Другими слова, если два запаздывающих требования обслуживаются не ранее чем с момента времени $t \geqslant \max\{r_i, r_j\}$, то при оптимальном расписании они могут быть обслужены в порядке неубывания директивных сроков.

Свойство 17. Если необходимо из двух требований j и i выбрать то, что будет обслуживаться с момента времени

$$\max\{r_i, r_j\} \leqslant t \leqslant \min\{d_j - p, d_i - p\},$$

а второе будет обслужено позже, то необходимо выбрать требование с наибольшим директивным сроком.

Свойство 18. Пусть при расписании $\pi = (\pi_1, j_1, j_2, j_3, j_4, \pi_2, j_5, \pi_3)$ *выполняется:*

$$S_{j_2}(\pi) < r_{j_5} < C_{j_2}(\pi), \quad C_{j_1} > r_{j_5} - p,\ S_{j_4} = r_{j_4} < r_{j_5} + 2p \text{ и } C_{j_5}(\pi) < d_{j_5}.$$

Тогда при расписании $\pi' = (\pi_1, j_5, j_3, j_4, \pi_2, j_1, j_2, \pi_3)$ *имеем* $F(\pi) \leqslant F(\pi')$.

См. рисунок, *ж*. Согласно этому свойству необходимо исключить представленную ситуацию перестановкой требования j_5 и требований j_2, j_3.

Свойство 19. Существует расписание, при котором любая попарная перестановка уменьшает значение целевой функции, но одновременное применение нескольких перестановок увеличивает значение целевой функции.

См. рисунок, *з*. При расписании $\pi = (4, 5, 6, 1, 2, 3, 7, 8, 9, 10)$ любая попарная перестановка,например, перестановки $4 \leftrightarrow 1$, $5 \leftrightarrow 2$, сокращает количество запаздывающих требований, но их одновременное применение $\pi = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$ увеличивает количество запаздывающих требований.

8. Заключение

Представлены свойства задачи и алгоритм решения задачи $1|r_j, p_j = p|\sum w_j T_j$. Вопрос о статусе вычислительной сложности задачи остается открытым. В том числе авторам не известны псевдополиномиальные алгоритмы решения или схемы приближенного решения с заданной относительной погрешностью.

Предполагаем, что задача $1|r_j, p_j = p|\sum w_j T_j$ является NP-трудной, но чтобы доказать это, необходим новый подход к доказательству NP-трудности.

Также в статье представлены новые задачи теории расписаний с обратными критериями оптимизации и их свойства.

СПИСОК ЛИТЕРАТУРЫ

1. *Graham R.L., Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G.* Optimization and Approximation in Deterministic Machine Scheduling: a Survey // Ann. Discret. Math. 1979. No. 5. P. 287–326.
2. *Leung J.Y.-T.* Handbook of Scheduling. N.Y.: Chapmann and Hall/CRC, 2004.
3. *Baptiste P.* Scheduling Equal-Length Jobs on Identical Parallel Machines // Discret. Appl. Math. 2000. No. 103(1-3). P. 21–32.
4. *Verma S., Dessouky M.* Single-Machine Scheduling of Unit-Time Jobs with Earliness and Tardiness Penalties // Math. Oper. Res. 1998. No. 23(4). P. 930–943.
5. *Van den Akker J.M., Diepen G., Hoogeveen J.A.* Minimizing Total Weighted Tardiness on a Single Machine with Release Dates and Equal-Length Jobs // J. Scheduling. 2010. No. 13. P. 561–576.
6. *Kravchenko S.A., Werner F.* Parallel Machine Problems with Equal Processing Times: a Survey // J. Scheduling. 2011. No. 14. P. 435–444.
7. *Held M., Karp R.M.* A Dynamic Programming Approach to Sequencing Problems // J. Soc. Indust. Appl. Math. 1962. No. 10. P. 196–210.

8. *Du J., Leung J.Y.-T.* Minimizing Total Tardiness on One Processor is NP-hard // Math. Oper. Res. 1990. No. 15. P. 483–495.
9. *Aloulou M.A., Kovalyov M.Y., Portmann M.-C.* Evaluation Flexible Solutions in Single Machine Scheduling via Objective Function Maximization: the Study of Computational Complexity // RAIRO Oper. Res. 2007. No. 41. P. 1–18.
10. *Gafarov E.R., Lazarev A.A., Werner F.* Transforming a Pseudo-Polynomial Algorithm for the Single Machine Total Tardiness Maximization // Probl. Polynom. One. Annal. Oper. Res. 2012. No. 196(1). P. 247–261.
11. *Gafarov E.R., Lazarev A.A., Werner F.* Single Machine Total Tardiness Maximization Problems: Complexity and Algorithms // Ann. Oper. Res. 2013. No. 207. P. 121–136.

Статья представлена к публикации членом редколлегии Ф.Т. Алескеровым.

Поступила в редакцию 07.07.2019

После доработки 03.11.2019

Принята к публикации 28.11.2019