Автоматика и телемеханика, № 10, 2010

© 2010 г. **Е. Р. ГАФАРОВ, канд. физ.-мат. наук,**
А. А. ЛАЗАРЕВ, д-р физ.-мат. наук
(Институт проблем управления им. В.А. Трапезникова РАН, Москва),
Ф. ВЕРНЕР, д-р философии
(Факультет математики университета Отто фон Герике,
Магдебург, Германия)

АЛГОРИТМЫ РЕШЕНИЯ ЗАДАЧ МАКСИМИЗАЦИИ СУММАРНОГО ЗАПАЗДЫВАНИЯ И МАКСИМИЗАЦИИ КОЛИЧЕСТВА ЗАПАЗДЫВАЮЩИХ ТРЕБОВАНИЙ ДЛЯ ОДНОГО ПРИБОРА[1]

Рассматриваются две одноприборные задачи теории расписаний максимизации суммарного запаздывания и максимизации количества запаздывающих требований, когда простои в обслуживании требований запрещены и требования начинают обслуживаться с момента времени 0. Показано, что задача максимизации количества запаздывающих требований полиномиально разрешима. Для некоторых частных случаев задачи максимизации суммарного запаздывания представлены точные полиномиальные алгоритмы решения, а также два точных алгоритма решения общего случая задачи.

1. Введение

Обычно в теории расписаний рассматриваются задачи, в которых необходимо минимизировать значение некоторой целевой функции. Например, популярным критерием оптимальности является минимизация общего времени завершения обслуживания всех требований. В качестве суммарных критериев оптимальности часто рассматривают минимизацию суммарного времени завершения обслуживания требований, суммарное запаздывание, количество запаздывающих требований. В данной работе рассматриваются две задачи с "обратными" критериями оптимальности, т.е. задачи максимизации суммарного запаздывания и максимизации количества запаздывающих требований для одного прибора.

Рассматриваемые задачи формулируются следующим образом. Необходимо обслужить n требований на одном приборе. Прерывания при обслуживании и обслуживание более одного требования в любой момент времени запрещены. Для требования $j \in N = \{1, \ldots, n\}$ 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 $\pi = (j_1, \ldots, j_n)$ строится с момента времени 0 и однозначно задаётся перестановкой элементов множества N. Обозначим через $C_{j_k}(\pi) = \sum_{l=1}^{k} p_{j_l}$ время завершения обслуживания требований j_k при расписании π. Если $C_j(\pi) > d_j$, тогда требование j запаздывает, и в этом случае полагают $U_j = 1$. Если $C_j(\pi) \leqslant d_j$, тогда требование j не запаздывает, и $U_j = 0$.

[1] Работа поддержана DAAD (Deutscher Akademischer Austauschdienst): A/08/80442/Ref. 325 и программами РАН № 15 и № 29.

Требуется построить расписание π^* обслуживания требований множества N, при котором достигается максимум функции $F(\pi) = \sum_{j=1}^{n} U_j(\pi)$. Обозначим данную задачу через $1||\max \sum U_j$.

Если $C_j(\pi) < d_j$, то выполнение требования j закончено раньше директивного срока. В этом случае полагают $V_j = 1$, иначе $V_j = 0$. Обозначим задачу минимизации количества требований, выполнение которых закончено раньше директивного срока, когда простои при обслуживании требований запрещены, через $1||\min \sum V_j$. Взаимосвязь задач $1||\max \sum U_j$ и $1||\min \sum V_j$ можно описать следующим образом. Пусть $d_j,\ p_j \in Z$ для всех $j \in N$ и δ есть вещественное число, $0 < \delta < 1$. Если обозначить $d'_j = d_j + \delta$, тогда задачи $1|d_j|\max \sum U_j$ и $1|d'_j|\min \sum V_j$ эквивалентны. Если требование запаздывает для задачи $1|d_j|\max \sum U_j$ при расписании π, тогда $C_j(\pi) > d_j$ и $C_j(\pi) > d_j + \delta = d'_j$, т.е. выполнение требования j закончено не раньше директивного срока.

Величина $T_j(\pi) = \max\{0, C_j(\pi) - d_j\}$ называется *запаздыванием* требования j при расписании π, а $F(\pi) = \sum_{j=1}^{n} T_j(\pi)$ – *суммарным запаздыванием* требований при расписании π. Обозначим $L_j(\pi) = C_j(\pi) - d_j$ и $E_j(\pi) = \max\{0, d_j - C_j(\pi)\}$. Задачу максимизации суммарного запаздывания, когда простои в обслуживании требований запрещены, будем обозначать через $1||\max \sum T_j$.

В данной работе запись вида $\{\pi\}$ обозначает множество требований, обслуживаемых при расписании π. Запись вида $i \in \pi$ означает $i \in \{\pi\}$. Через $\pi \setminus \{i\}$, где $\pi = (\pi_1, i, \pi_2)$, π_1 и π_2 – частичные расписания, будем обозначать частичное расписание (π_1, π_2). Запись $j \to i$ означает, что обслуживание требования j предшествует обслуживанию требования i. Соответственно, запись $(j \to i)_\pi$ означает, что это выполняется при расписании π.

Известно, что "исходная" задача $1||\min \sum U_j$ полиномиально разрешима за время $O(n \log n)$ при помощи алгоритма Мура [1]. Задача $1\,||\,\min \sum T_j$ является NP-трудной в обычном смысле [2, 3]. Лаулер [4] предложил точный псевдополиномиальный алгоритм трудоемкости $O(n^4 \sum p_j)$ операций. Модификция алгоритма [5] позволяет находить решение для примеров размерности до $n \leqslant 500$, построенных по схеме [6]. В работе [7] показано, что на двух специальных классах примеров подобный алгоритм может решать только примеры гораздо меньшей размерности, чем $n = 500$.

Для NP-трудной в сильном смысле задачи $1|r_j|L_{\max}$ показано [8], что "обратная" задача $1|r_j|\max\ f_{\min}$ полиномиально разрешима за время $O(n^2)$ для произвольных неубывающих функций $f_j(t),\ \forall j \in N$.

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, при помощи алгоритма решения задачи $1||\max \sum U_j$ можно вычислить максимальное количества запаздывающих требований, что, в свою очередь, позволяет сократить перебор в алгоритме решения задачи максимизации суммарного запаздывания $1||\max \sum T_j$ (см. раздел 3 статьи). 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. Более того, существует связь между классическими задачами минимизации и "обратны-

ми" задачами. К примеру, две задачи $1||\max\sum U_j$ и $1||\min\sum V_j$, когда простои в обслуживании требований запрещены, эквивалентны.

С другой стороны, для данных проблем существуют практические интерпретации и приложения. Например, монтажная команда должна смонтировать ветряные электрогенераторы (турбины) в разных районах страны. В каждом районе j необходимо смонтировать определенное 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 $1||\max\sum T_j$.

Статья имеет следующую структуру. Во втором разделе представлен полиномиальный алгоритм решения задачи $1||\max\sum U_j$. В третьем разделе рассматривается задача максимизации суммарного запаздывания для одного прибора $1||\max\sum T_j$. В подразделе 3.1 представлены правила сокращения перебора Эммонса. В следующем подразделе ПРИВОДИТСЯ обзор полиномиально разрешимых и NP-трудных случаев для "исходной" и "обратной" задач. В подразделе 3.3 ОПИСАНЫ некоторые свойства оптимальных расписаний для задачи $1||\max\sum T_j$. Точные полиномиальные алгоритмы решения четырех частных случаев задачи приводятся в подразделах 3.4–3.7. В подразделе 3.8 представлен точный псевдополиномиальный алгоритм решения общего случая задачи. Альтернативный точный алгоритм приводится в подразделе 3.9.

2. Полиномиальный алгоритм решения задачи $1||\max\sum U_j$

В данном разделе представлены некоторые свойства оптимальных расписаний для рассматриваемой задачи, а также приводится точный алгоритм решения.

Лемма 1. Для каждого примера задачи существует оптимальное расписание вида $\pi = (S, F)$, при котором все требования $j \in F$ запаздывают, а все требования $i \in S$ не запаздывают.

Доказательство. Предположим, что существует оптимальное расписание вида $\pi = (j_1 \ldots, j_{k-1}, j_k, \ldots, j_n)$, при котором требование j_{k-1} запаздывает, а требование j_k не запаздывает. Тогда $\pi' = (j_1, \ldots, j_k, j_{k-1}, \ldots, j_n)$ также является оптимальным расписанием. Производя данную попарную перестановку, получим оптимальное расписание вида $\pi = (S, F)$.

Лемма 2. Для каждого примера задачи существует оптимальное расписание вида $\pi = (S, F)$, при котором для любой пары требований $j \in F$ и $i \in S$ неравенства $p_j > p_i$ и $d_j \geqslant d_i$ не выполняются одновременно.

Доказательство. Предположим, что существует оптимальное расписание вида $\pi = (j_1, \ldots, j_k, j_{k+1}, \ldots, j_{l-1}, j_l, \ldots, j_n)$, где $p_{j_l} > p_{j_k}, d_{j_l} \geqslant d_{j_k}$, $C_{j_l}(\pi) > d_{j_l}$ и $C_{j_k}(\pi) \leqslant d_{j_k}$. Тогда расписание $\pi' = (j_1, \ldots, j_l, j_{k+1} \ldots, j_{l-1}, j_k, \ldots, j_n)$ также оптимальное, так как $C_{j_k}(\pi') = C_{j_l}(\pi) > d_{j_l} \geqslant d_{j_k}$ и для всех требований $j_i = j_{k+1}, \ldots$ $\ldots, j_{l-1}$ выполняется $C_{j_i}(\pi') > C_{j_i}(\pi)$, так как $p_{j_l} > p_{j_k}$.

Пусть d_j, $p_j \in Z$ для всех $j \in N$ и δ есть вещественное число, $0 < \delta < 1$. Вычислим значения $r_j = d_j - p_j + \delta$ для каждого требования $j \in N$.

Лемма 3. Для каждого примера задачи существует оптимальное расписание вида $\pi = (S, F)$, при котором все запаздывающие требования $j_{k+1}, j_{k+2}, \ldots, j_n$ обслуживаются в порядке $r_{j_{k+1}} \leqslant r_{j_{k+2}} \leqslant \ldots \leqslant r_{j_n}$.

Доказательство. Предположим, что существует оптимальное расписание вида $\pi = (\pi_1, i, j, \pi_2)$, где требования i и j запаздывают, но $r_j < r_i$. Очевидно, что если требование i запаздывает, тогда $C_i(\pi) - p_i \geqslant r_i > r_j$. Рассмотрим расписание $\pi' = (\pi_1, j, i, \pi_2)$. Имеем $C_j(\pi') - p_j = C_i(\pi) - p_i \geqslant r_i > r_j$ и $C_i(\pi') = C_i(\pi) + p_j > d_i + p_j > d_i$. Поэтому расписание π' является оптимальным, причем оба требования i и j по-прежнему запаздывают.

Следующий алгоритм является точным полиномиальным алгоритмом решения задачи.

Алгоритм 1.

Шаг 1. $S := \emptyset; F := N$.

Шаг 2. Построим расписание вида $\pi = \left(j_1^s, \ldots, j_k^s, j_{k+1}^f, j_{k+2}^f, \ldots, j_n^f\right)$. Все требования из множества $S = \left\{j_1^s, \ldots, j_k^s\right\}$ обслуживаются в начале расписания. Все требования из множества $F = \left\{j_{k+1}^f, j_{k+2}^f, \ldots, j_n^f\right\}$ обслуживаются в порядке $r_{j_{k+1}^f} \leqslant r_{j_{k+2}^f} \leqslant \ldots \leqslant r_{j_n^f}$.

Шаг 3. Пусть требование $j_l^f \in F$ – последнее незапаздывающие требование. Если такого требования нет, тогда ПЕРЕЙТИ на шаг 6.

Шаг 4. Выберем требование $j^* \in \left\{j_l^f, \ldots, j_n^f\right\}$ с максимальной продолжительностью обслуживания: $p_{j^*} \geqslant p_i$ для всех $i \in \left\{j_l^f, \ldots, j_n^f\right\}$.

Шаг 5. $S := S \cup \{j^*\}$, $F := F \setminus \{j^*\}$, ПЕРЕЙТИ на шаг 2.

Шаг 6. Оптимальное расписание $\pi = \left(j_1^s, \ldots, j_k^s, j_{k+1}^f, j_{k+2}^f, \ldots, j_n^f\right)$ получено.
КОНЕЦ АЛГОРИТМА.

Лемма 4. Пусть при расписании $\pi = (j_1, \ldots, j_l, j_{l+1}, \ldots, j_n)$, $r_{j_1} \leqslant r_{j_2} \leqslant \ldots \leqslant r_{j_n}$, требование j_l – последнее незапаздывающее требование. Тогда существует оптимальное расписание, при котором требование $j^* \in \{j_l, j_{l+1}, \ldots, j_n\}$, $p_{j^*} \geqslant p_i$, для всех $i \in \{j_l, j_{l+1}, \ldots, j_n\}$, не запаздывает.

Доказательство. Предположим, что существует оптимальное расписание $\pi = (\pi_1, i, \pi_2, j^*, \pi_3, \pi_4)$, при котором требование j^* запаздывает. Очевидно, что не существует расписания, при котором все требования $j_l, j_{l+1}, \ldots, j_n$ запаздывают (см. лемму 3). Тогда существует незапаздывающее требование $i \in \{j_l, j_{l+1}, \ldots, j_n\}$, $i \neq j^*$. В соответствии с леммой 3 пусть все требования $\{j^*, \pi_3, \pi_4\}$ обслуживаются в порядке неубывания значений r_j.

Если $r_{j^*} \geqslant r_i$, тогда $r_{j^*} + p_{j^*} \geqslant r_i + p_i$, т.е. $d_{j^*} \geqslant d_i$, и расписание $\pi' = (\pi_1, j^*, \pi_2, i, \pi_3, \pi_4)$ также оптимальное. При расписании π' требование j^* не запаздывает, а требование i запаздывает.

Рассмотрим случай $r_{j^*} < r_i$. Пусть для всех $j \in \{\pi_4\}$ $r_j \geqslant r_i$, и для всех $j \in \{\pi_3\}$ выполняется $r_j < r_i$. Рассмотрим расписание $\pi'' = (\pi_1, j^*, \pi_2, \pi_3, i, \pi_4)$. Для всех $j \in \{\pi_2\}$ имеем $C_j(\pi'') \geqslant C_j(\pi)$, так как $p_{j^*} \geqslant p_i$. При расписании π'' все требования из множества $\{\pi_3\} \bigcup \{i\} \bigcup \{\pi_4\}$ запаздывают, так как при расписании $\pi''' = (\pi_1, \pi_2, j^*, \pi_3, i, \pi_4)$ только требование j^* может быть незапаздывающим в соответствии с условием леммы. Поэтому расписание π'' оптимальное.

Лемма 5. При помощи алгоритма 1 для задачи $1||\max \sum U_j$ за $O(n^2)$ операций строится оптимальное расписание.

Доказательство. Докажем лемму методом математической индукции. Алгоритм верен при $n = 1$.

Предположим, что алгоритм верен для всех примеров размерности $n-1$ требований. Рассмотрим пример размерности n требований. Предположим, что при помощи

алгоритма 1 было построено расписание $\pi = (S, F)$, но существует оптимальное расписание $\pi' = (S', F')$, что $|F'| \geqslant |F|$. При использовании данного алгоритма для $n-1$ требований $1, 2, \ldots, j-1, j+1, \ldots, n$, где $j = j^*$, будет получено оптимальное расписание $(S \setminus \{j^*\}, F)$. Расписание $(S' \setminus \{j^*\}, F')$ является допустимым для соответствующего примера из $n-1$ требований. Поэтому имеем $|F'| \leqslant |F|$, тогда $|F'| = |F|$.

Подобный алгоритм для задачи $1||\min \sum V_j$ впервые был представлен в [9], к сожалению, без доказательства.

3. Алгоритмы решения задачи $1||\max \sum T_j$

В данном разделе рассматривается задача максимизации суммарного запаздывания, т.е. функции $F(\pi) = \sum\limits_{j=1}^{n} \max\{0, C_j(\pi) - d_j\}$. Прерывания при обслуживании требований запрещены, требования начинают обслуживаться в момент времени 0.

Ниже приводятся некоторые правила сокращения перебора для данной задачи, а также точные алгоритмы решения частных случаев задачи.

3.1. Правила сокращения перебора Эммонса

Для классической задачи минимизации суммарного запаздывания Эммонс предложил правила сокращения перебора [5], названные впоследствии его именем. Далее приводятся правила сокращения перебора, адаптированные к задаче максимизации.

Пусть B_j – множество предшественников требования j при всех оптимальных расписаниях, и A_j – множество последователей требования j при всех оптимальных расписаниях, т.е. $B_j = \{i \in N : i \to j$ при всех оптимальных расписаниях$\}$ и $A_j = \{i \in N : j \to i$ при всех оптимальных расписаниях$\}$. Определим $c_j = \sum\limits_{k=1}^{n} p_k - \sum\limits_{i \in A_j} p_i$ и $s_j = \sum\limits_{i \in B_j} p_i$. Тогда выполняется $s_j + p_j \leqslant C_j \leqslant c_j$ при всех оптимальных расписаниях.

Лемма 6. Если $p_j \geqslant p_i$ и $d_j \geqslant d_i$, или $p_j \geqslant p_i$, $d_j < d_i$ и $d_j \geqslant c_j$, или $p_j \geqslant p_i$, $d_j < d_i$ и $d_i \leqslant s_i + p_j$, тогда существует оптимальное расписание π, при котором $(j \to i)_\pi$.

Доказательство. Рассмотрим каждое из трех условий отдельно.

а) Пусть $p_j \geqslant p_i$ и $d_j \geqslant d_i$. Предположим, что существует оптимальное расписание $\pi = (\pi_1, i, \pi_2, j, \pi_3)$. Для расписания $\pi' = (\pi_1, j, \pi_2, i, \pi_3)$ выполняется $F(\pi') - F(\pi) \geqslant (T_j(\pi') - T_j(\pi)) + (T_i(\pi') - T_i(\pi))$. Если $C_j(\pi') > d_j$, тогда $F(\pi') - F(\pi) \geqslant -\left(p_i + \sum\limits_{k \in \pi_2} p_k\right) + \left(p_i + \sum\limits_{k \in \pi_2} p_k\right) = 0$. Иначе, если $C_j(\pi') \leqslant d_j$, выполняется $F(\pi') - F(\pi) \geqslant -\max\{0, C_j(\pi) - d_j\} + \max\{0, C_j(\pi) - d_i\} \geqslant 0$.

б) Пусть $p_j \geqslant p_i$, $d_j < d_i$ и $d_j \geqslant c_j$. Предположим, что существует оптимальное расписание $\pi = (\pi_1, i, \pi_2, j, \pi_3)$. Для расписания $\pi' = (\pi_1, j, i, \pi_2, \pi_3)$ выполняется $F(\pi') - F(\pi) \geqslant (T_j(\pi') - T_j(\pi)) + (T_i(\pi') - T_i(\pi)) = (0-0) + (0-0) = 0$.

в) Пусть $p_j \geqslant p_i$, $d_j < d_i$ и $d_i \leqslant s_i + p_j$. Предположим, что существует оптимальное расписание $\pi = (\pi_1, i, \pi_2, j, \pi_3)$. Для расписания $\pi' = (\pi_1, j, \pi_2, i, \pi_3)$ выполняется

$$F(\pi') - F(\pi) \geqslant (T_j(\pi') - T_j(\pi)) + (T_i(\pi') - T_i(\pi)) \geqslant \left(-p_i - \sum_{k \in \pi_2} p_k\right) + \left(\sum_{k \in \pi_2} p_k + p_i\right) = 0.$$

Во всех случаях расписание π' также оптимальное.

Таблица 1. Правила сокращения перебора Эммонса

$p_j \geqslant p_i$	$1\|\|\min \sum T_j$	$p_j \geqslant p_i$	$1\|\|\max \sum T_j$
$(i \to j)$	$d_i \leqslant d_j$ или $d_i \leqslant s_j + p_j$	$(j \to i)$	$d_i \leqslant d_j$ или $(d_i > d_j$ и $d_j > c_j)$ или $(d_i > d_j$ и $d_i < s_i + p_i)$ или $(d_i > d_j$ и $d_i < s_i + p_j)$
$(j \to i)$	$d_i + p_i \geqslant c_j$ и $(d_i > d_j$ или $d_i > s_j + p_j)$	$(i \to j)$	$d_i > c_i - p_i$ и $d_j \leqslant s_j + p_j$

Лемма 7. Если $p_j \geqslant p_i$ и $d_i > c_i - p_i$ и $d_j \leqslant s_j + p_j$, тогда существует оптимальное расписание π, при котором $(i \to i)_\pi$.

Доказательство. Предположим, что существует оптимальное расписание $\pi = (\pi_1, j, \pi_2, i, \pi_3)$. Для расписания $\pi' = (\pi_1, i, j, \pi_2, \pi_3)$ выполняется $F(\pi') - F(\pi) \geqslant$ $\geqslant (T_j(\pi') - T_j(\pi)) + (T_i(\pi') - T_i(\pi)) = p_i - \max\{c_i - d_i, 0\} > 0$.

В табл. 1 приводится сравнение правил сокращения перебор для задач минимизации и максимизации суммарного запаздывания.

3.2. Частные случаи задачи максимизации суммарного запаздывания

Некоторые полиномиально разрешимые случаи задачи минимизации суммарного запаздывания рассматриваются в [10]. В табл. 2 приведен обзор результатов, полученных для частных случаев задачи максимизации, и сравнение с результатами, полученными для задачи минимизации.

Таблица 2. Полиномиально разрешимые и NP-трудные частные случаи

Частный случай	$1\|\|\min \sum T_j$		$1\|\|\max \sum T_j$	
	P/NP	Алгоритм	P/NP	Алгоритм
$d_j = d$	P	SPT	P	LPT *Лемма 6*
$p_j = p$	P	EDD	P	$\pi_{\text{opt}} = (1, 2, \ldots, n)$, $d_1 \geqslant d_2 \geqslant \ldots \geqslant d_n$ *Лемма 6*
$d_1 \leqslant d_2 \leqslant \ldots \leqslant d_n$, $p_1 \leqslant p_2 \leqslant \ldots \leqslant p_n$.	P	SPT	P	LPT *Лемма 6*
$d_{\max} - d_{\min} \leqslant 1,\ p_j \in Z$	P	$O(n^2)$ [10, 11]	P	$O(n^3)$ *Лемма 18*
$d_1 \leqslant d_2 \leqslant \ldots \leqslant d_n$, $d_i - d_{i-1} > p_i, i = 2, \ldots, n$.	P	$O(n^2)$ [10, 11]	откр.	-
Canonical LG [3] $d_1 \leqslant d_2 \leqslant \ldots \leqslant d_n$, $p_1 \geqslant p_2 \geqslant \ldots \geqslant p_n$, $d_n - d_1 \leqslant p_n,\ \ldots$ (см. (LG) в 2.5)	NP-труд. [3]	$O(np_{\min})$ [10, 11]	P	$O(n^5)$ *Лемма 20*
Canonical DL [2]	NP-труд. [2]	$O(n \sum p_j)$ [10, 11]	P	$O(n^5)$ *Лемма 21*
$d_1 + p_1 \leqslant d_2 + p_2 \leqslant \ldots \leqslant d_n + p_n$, $p_1 < p_2 < \cdots < p_n$.	P	$O(n^3)$	P	$O(n^2)$ *Лемма 22*

3.3. Свойства оптимальных расписаний для задачи $1||\max\sum T_j$

В данном подразделе представлены некоторые свойства оптимальных расписаний для задачи максимизации суммарного запаздывания.

Лемма 8. Для каждого примера задачи существует оптимальное расписание вида $\pi = (S, F) = (SPT, LPT)$*, где все требования* $j \in F$ *запаздывают, а все требования* $i \in S$ *не запаздывают. Все требования из множества* S *обслуживаются в порядке неубывания продолжительности обслуживания (SPT), а все требования из множества* F *обслуживаются в порядке невозрастания продолжительности обслуживания (LPT).*

Доказательство.

1. Предположим, что существует оптимальное расписание $\pi = (\pi_1, j, \pi_2, i, \pi_3)$, где требование $j \in F$ запаздывает, а требование $i \in S$ не запаздывает. Для расписания $\pi' = (\pi_1, i, j, \pi_2, \pi_3)$ выполняется: $F(\pi') - F(\pi) \geqslant (T_j(\pi') - T_j(\pi)) + (T_i(\pi') - -T_i(\pi)) = (p_i) + (0) > 0$. Получили противоречие, так как для расписания π' значение целевой функции больше, а значит расписание π не оптимальное.

2. Рассмотрим оптимальное расписание вида $\pi = (S, F)$, где все требования $j \in F$ запаздывают, а все требования $i \in S$ не запаздывают. Покажем, что все требования $j \in F$ обслуживаются в LPT порядке (по невозрастанию продолжительности обслуживания требований). Предположим, что существует оптимальное расписание $\pi = (\pi_1, j_1, j_2, \pi_2)$, при котором требования j_1 и j_2 запаздывают и $p_{j_1} < p_{j_2}$. Для расписания $\pi' = (\pi_1, j_2, j_1, \pi_2)$, выполняется $F(\pi') - F(\pi) = (T_{j_1}(\pi') - T_{j_1}(\pi)) + (T_{j_2}(\pi') - -T_{j_2}(\pi)) \geqslant p_{j_2} - \min\{p_{j_1}, T_{j_2}(\pi)\} > 0$. Получили противоречие, а значит, расписание $\pi = (\pi_1, j_1, j_2, \pi_2)$ не оптимальное.

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

Пусть $U_{\max}$ – максимальное количество запаздывающих требований для примера задачи. Тогда $n - U_{\max}$ минимальное количество незапаздывающих требований. Значение $U_{\max}$ может быть рассчитано алгоритмом 1 трудоемкости $O(n^2)$ операций.

Пусть N_1 – множество $n - U_{\max} - 1$ требований с минимальной продолжительностью обслуживания и N_2 – множество $n - U_{\max}$ требований с минимальной продолжительностью обслуживания. Определим N_i как множество $N_1 \bigcup \{i\}$, если $i \notin N_1$, иначе $N_i = N_2$.

Лемма 9. Если $d_i < \sum\limits_{k \in N_i} p_k$*, тогда не существует оптимального расписания, при котором требование* i *не запаздывает.*

Доказательство. Докажем лемму от противного. Предположим, что существует оптимальное расписание $\pi = (S, F)$, где все требования $j \in F$ запаздывают, а все требования $i \in S$ не запаздывают. Пусть для требования $i \in S$ выполняется $d_i < \sum\limits_{k \in N_i} p_k$. Тогда $d_i < \sum\limits_{k \in N_i} p_k \leqslant \sum\limits_{k \in S} p_k$, и расписание $\pi' = (S \setminus \{i\}, i, F)$ лучше, т.е. $F(\pi') > F(\pi)$. В данном расписании требование i запаздывает.

Лемма 10. Если требование j запаздывает при оптимальном расписании, тогда все требования $i \in N,\ p_i < p_j,\ d_i \leqslant d_j$ и требования $i \in N,\ p_i = p_j,\ d_i < d_j$ также запаздывают при данном расписании.

Доказательство. Лемма может быть доказана аналогично доказательству леммы 6, т.е. если требование $i \in N,\ p_i < p_j,\ d_i \leqslant d_j$ не запаздывает, тогда можно поменять местами требования j и i в расписании (в перестановке).

Лемма 11. Если для требования j выполняется

$$\left(\sum_{i=1}^{n} p_i - \sum_{k \in \{i \in N,\ p_i < p_j,\ d_i \leqslant d_j\} \bigcup \{i \in N,\ p_i = p_j,\ d_i < d_j\}} p_k\right) \leqslant d_j,$$

тогда не существует оптимального расписания, где требование j запаздывает.

Доказательство. Доказательство леммы следует из леммы 10. Если требование j запаздывает при оптимальном расписании, тогда все требования из множества $\{i \in N,\ p_i < p_j,\ d_i \leqslant d_j\} \bigcup \{i \in N,\ p_i = p_j,\ d_i < d_j\}$ также запаздывают и обслуживаются после требования j, так как при оптимальном расписании все запаздывающие требования обслуживаются в порядке LPT. Тогда при любом оптимальном расписании π выполняется $C_j(\pi) \leqslant \sum\limits_{i=1}^{n} p_i - \sum\limits_{k \in \{i \in N,\ p_i < p_j,\ d_i \leqslant d_j\} \bigcup \{i \in N,\ p_i = p_j,\ d_i < d_j\}} p_k$ и должно выполняться $C_j(\pi) > d_j$, если требование j запаздывает.

Лемма 12. Пусть j – первое запаздывающее требование при оптимальном расписании $\pi = (S, j, F)$. Тогда для всех требований $i \in S$ выполняется $d_i \geqslant \geqslant \max\{C_j(\pi) - p_j, d_j\}$.

Доказательство. Если для требования $i \in S$ выполняется $d_i < C_j(\pi) - p_j$, тогда требование i запаздывает при расписании $\pi' = (S \setminus \{i\}, i, j, F)$, и $F(\pi') > F(\pi)$. Если $d_i < d_j$, тогда для расписания $\pi' = (S \setminus \{i\}, j, i, F)$ имеем $F(\pi') > F(\pi)$.

Лемма 13. Пусть j — запаздывающее требование при расписании $\pi = (\pi_1, \pi_2, j, \pi_3)$, при котором все требования из π_2 запаздывают и $|\{\pi_2\}| = k > 0$. Тогда $T_j(\pi) \geqslant kp_j$.

Доказательство. Если $T_j(\pi) < kp_j$, тогда для расписания $\pi = (j, \pi_1, \pi_2, \pi_3)$ выполняется $F(\pi') > F(\pi)$.

Перенумеруем требования согласно правилу: $d_1 \leqslant d_2 \leqslant \ldots \leqslant d_n$, если $d_i = d_{i+1}$, тогда $p_i \leqslant p_{i+1}$. Пусть j – требование с максимальной продолжительностью обслуживания, $j := \operatorname{argmax}\left\{d_i : p_i = \max\limits_{k \in N} p_k\right\}$.

Лемма 14. Существует оптимальное расписание, где первое запаздывающее требование принадлежит множеству $\{1, 2, \ldots, j\}$. Если $d_j < d_{j+1}$, тогда при всех оптимальных расписаниях первое запаздывающее требование принадлежит множеству $\{1, 2, \ldots, j\}$.

Доказательство. Проведем доказательство от противного. Пусть существует оптимальное расписание $\pi = (\pi_1, i, \pi_2)$, где $i > j$ и требование i – первое запаздывающее. Тогда j не запаздывает в данном оптимальном расписании, так как все запаздывающие требования обслуживаются в порядке LPT при любом оптимальном расписании. Пусть $\pi = (j, \pi_1, i, \pi_2)$. Тогда для расписания $\pi' = (\pi_1, i, j, \pi_2)$ выполняется $F(\pi') = F(\pi)$, если $d_j = d_i$, и $F(\pi') > F(\pi)$, если $d_j < d_i$.

Пусть $M_l = \{i \in N, p_i > p_l\}$ и $O_l = \{i \in N, i > l\}$.

Лемма 15. Если требование l – первое запаздывающее требование при расписании π, тогда для каждого требования $i \in M_l$ выполняется $\sum\limits_{k \in M_l} p_k \leqslant d_i$.

Доказательство очевидно, так как все запаздывающие требования при оптимальном расписании обслуживаются в порядке LPT, т.е. все требования $i \in M_l$ запаздывают.

Лемма 16. Если требование l – первое запаздывающее требование при расписании π и $l = \operatorname{argmax}\{d_i : p_i = \max\limits_{k \in N} p_k\}$, тогда $\sum\limits_{k \in O_l} p_k + p_l > d_l$.

Леммы 9–16 могут быть использованы для сокращения перебора в алгоритмах решения задачи.

3.4. Точный алгоритм решения частного случая $d_{\max} - d_{\min} \leqslant 1$

Определим $z = \lceil d_{\max} \rceil$. Следующий алгоритм 2 решения является точным для рассматриваемого частного случая.

Алгоритм 2.

Шаг 1. $S := N,\ F := \emptyset,\ P_f := 0$.

Шаг 2. Если существует требование $j \in S$, для которого $\sum\limits_{i=1}^{n} p_i - P_f - p_j \geqslant z + 1$, тогда ПЕРЕЙТИ на шаг 3 алгоритма, иначе ПЕРЕЙТИ на шаг 5.

Шаг 3. Выберем требование $j^*(S) \in S$ с минимальной продолжительностью обслуживания p_j. Если существует несколько таких требований, выберем среди них требование с минимальным директивным сроком d_j:

$$j^*(S) = \arg\min_{j \in S} \left\{ d_j : p_j = \min_{i \in S} p_i \right\}.$$

Шаг 4. $S := S \setminus \{j^*(S)\}$, $F := F \bigcup \{j^*(S)\}$, $P_f := P_f + p_{j^*(S)}$, ПЕРЕЙТИ на шаг 2.

Шаг 5. Теперь можно дополнительно выбрать только три запаздывающих требования. Для каждой тройки требований $j_1,\ j_2, j_3 \in S$ рассматриваем все расписания $\pi' = (\pi_1, \pi^{123}, \pi_2)$, где $\{\pi_1\} = S \setminus \{j_1, j_2, j_3\}$, $\{\pi_2\} = F$, $\{\pi^{123}\} = \{j_1, j_2, j_3\}$ и все требования из множества F обслуживаются в порядке LPT. После выберем лучшее расписание (с максимальным значением целевой функции) среди $|S|(|S|-1)(|S|-2)$ рассмотренных расписаний.

Лемма 17. Если $\sum\limits_{i=1}^{n} p_i - p_{j^*(N)} \geqslant z + 1$, тогда существует оптимальное расписание, при котором требование $j^*(N)$ запаздывает.

Доказательство. Обозначим $j^* = j^*(N)$. Пусть существует оптимальное расписание $\pi = (\pi_1, j^*, \pi_2, \alpha, i)$. Если $p_{j^*} = p_i, d_{j^*} \leqslant d_i$, тогда расписание $\pi' = (\pi_1, i, \pi_2, \alpha, j^*)$ также оптимально.

Если $p_{j^*} < p_i$, тогда требование j^* не запаздывает при расписании π. Для расписания $\pi' = (\pi_1, i, \pi_2, \alpha, j^*)$ выполняется:

а) Если $C_i(\pi') \leqslant d_i$, тогда $T_i(\pi') - T_i(\pi) = -\sum\limits_{k=1}^{n} p_k + d_i$, $T_{j^*}(\pi') - T_{j^*}(\pi) = \sum\limits_{k=1}^{n} p_k - d_{j^*}$, $T_\alpha(\pi') - T_\alpha(\pi) \geqslant 1$, так как $p_i \geqslant p_{j^*} + 1$ ($p_j \in Z$ для всех $j \in N$).
Тогда $F(\pi') - F(\pi) \geqslant (T_{j^*}(\pi') - T_{j^*}(\pi)) + (T_i(\pi') - T_i(\pi)) + (T_\alpha(\pi') - T_\alpha(\pi)) \geqslant 0$, так как $d_{\max} - d_{\min} \leqslant 1$.

б) Если $C_i(\pi') > d_i$, тогда $T_i(\pi') - T_i(\pi) = -\sum\limits_{k \in \pi_2 \bigcup \{\alpha, j^*\}} p_k$, $T_{j^*}(\pi') - T_{j^*}(\pi) \geqslant \geqslant \sum\limits_{k \in \pi_2 \bigcup \{\alpha, j^*\}} p_k - 1$, так как $d_{\max} - d_{\min} \leqslant 1$.
Имеем $T_\alpha(\pi') - T_\alpha(\pi) \geqslant 1$, так как $p_i \geqslant p_{j^*} + 1$ ($p_j \in Z$ для всех $j \in N$). Тогда $F(\pi') - F(\pi) \geqslant 0$.

Лемма 18. При помощи алгоритма 2 для частного случая $d_{\max} - d_{\min} \leqslant 1$ *за время* $O(n^3)$ *строится оптимальное расписание.*

Доказательство. Оптимальность шагов 3–4 алгоритма устанавливается леммой 17. Трудоемкость шага 5 равна $O(n^3)$ операций.

3.5. Точный алгоритм решения частного случая "Canonical LG"

Рассматривается следующий частный случай задачи (см. [3]):

$$
\text{(LG)} \quad \begin{cases}
p_1 > p_2 > \cdots > p_{2n+1}, \\
d_1 < d_2 < \cdots < d_{2n+1}, \\
d_{2n+1} - d_1 < p_{2n+1}, \\
p_{2n+1} = n^3 b, \\
p_{2n} = p_{2n+1} + b, \\
p_{2i} = p_{2i+2} + b, \ i = n-1, \ldots, 1, \\
p_{2i-1} = p_{2i} + \delta_i, \ i = n, \ldots, 1, \\
d_{2n+1} = \sum\limits_{i=1}^{n} p_{2i} + p_{2n+1} + \frac{1}{2}\delta, \\
d_{2n} = d_{2n+1} - \delta, \\
d_{2i} = d_{2i+2} - (n-i)b + \delta, \ i = n-1, \ldots, 1, \\
d_{2i-1} = d_{2i} - (n-i)\delta_i - \varepsilon\delta_i, \ i = n, \ldots, 1,
\end{cases}
$$

где $\delta_1, \delta_2, \ldots, \delta_n \in Z$, $\delta = \sum\limits_{i=1}^{n} \delta_i$, $b = n^2\delta$, $0 < \varepsilon < \dfrac{\min\limits_i \delta_i}{\max\limits_i \delta_i}$.

Обозначим

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

Лемма 19 [3]. *Для данного частного случая LG при каждом расписании запаздывает* n *или* $n+1$ *требований.*

Лемма 20. Для частного случая LG оптимальное расписание может быть рассчитано за $O(n^5)$ *операций.*

Доказательство. Здесь приводится только общая схема доказательства (полное доказательство приводится в [12]). Без ограничения общности примем $n \geqslant 2$. Сначала доказываем, что требование V_{2n+1} обслуживается на последней позиции в любом оптимальном расписании. Далее показываем, что на последних позициях расписания обслуживаются $\left(\left\lfloor \dfrac{n}{2} \right\rfloor - 1\right)$ требований с минимальной продолжительностью обслуживания. Далее можно доказать, что первые запаздывающие $n - \left(\left\lfloor \dfrac{n}{2} \right\rfloor + 2\right)$ требований при всех оптимальных расписаниях есть требования с максимальной продолжительностью обслуживания (и, соответственно, минимальным директивным сроком). То есть известно по крайней мере $n-3$ запаздывающих требований при любом оптимальном расписании. Поэтому необходимо дополнительно выбрать еще не более 4 запаздывающих требований, так как при любом расписании запаздывает не более чем n или $n+1$ требований. Следовательно необходимо рассмотреть не более $O(n^4)$ расписаний. Данную операцию можно выполнить за время $O(n^5)$.

3.6. Точный алгоритм решения частного случая задачи "Canonical DL" [2]

Для данного частного случая задано $3\bar{n}+1$ требований:

$$N=\{V_1,V_2,\ldots,V_{2i-1},V_{2i},\ldots,V_{2\bar{n}-1},V_{2\bar{n}},W_1,W_2,\ldots,W_{\bar{n}+1}\}.$$

Задано $b_1\geqslant b_2\geqslant\ldots\geqslant b_{2\bar{n}}$, $b_i\in Z$, $i=1,2,\ldots,2\bar{n}$, $\delta=\dfrac{1}{2}\sum\limits_{i=1}^{\bar{n}}(b_{2i-1}-b_{2i})$ и $b=(4\bar{n}+1)\delta$.

Кроме того,

$$\begin{aligned}
&a_1=b_1+(9\bar{n}^2+3\bar{n})\delta+5\bar{n}(b_1-b_{2\bar{n}}),\\
&a_2=b_2+(9\bar{n}^2+3\bar{n})\delta+5\bar{n}(b_1-b_{2\bar{n}}),\\
&\ldots,\\
&a_{2i-1}=b_{2i-1}+(9\bar{n}^2+3\bar{n}-i+1)\delta+5\bar{n}(b_1-b_{2\bar{n}}),\\
&a_{2i}=b_{2i}+(9\bar{n}^2+3\bar{n}-i+1)\delta+5\bar{n}(b_1-b_{2\bar{n}}),\\
&\ldots,\\
&a_{2n-1}=b_{2n-1}+(9\bar{n}^2+2\bar{n}+1)\delta+5\bar{n}(b_1-b_{2\bar{n}}),\\
&a_{2n}=b_{2n}+(9\bar{n}^2+2\bar{n}+1)\delta+5\bar{n}(b_1-b_{2\bar{n}}).
\end{aligned}$$

$$\begin{aligned}
&p_{V_i}=a_i, \qquad 1\leqslant i\leqslant 2\bar{n};\\
&p_{W_i}=b, \qquad 1\leqslant i\leqslant \bar{n}+1;\\
&d_{V_i}=\begin{cases}(j-1)b+\delta+(a_2+a_4+\cdots+a_{2j}), & \text{если } i=2j-1,\\ d_{V_{2j-1}}+2(\bar{n}-j+1)(a_{2j-1}-a_{2j}), & \text{если } i=2j,\end{cases}\\
&d_{W_i}=\begin{cases}ib+(a_2+a_4+\cdots+a_{2i}), & 1\leqslant i\leqslant\bar{n},\\ d_{W_{\bar{n}}}+\delta+b, & i=\bar{n}+1.\end{cases}
\end{aligned}$$

Лемма 21. Для частного случая DL оптимальное расписание может быть найдено за $O(n^5)$ операций.

Доказательство. Здесь приводится только общая схема доказательства (полное доказательство см. в [12]). Без ограничения общности, пусть $\bar{n}\geqslant 3$. Очевидно, что при любом расписании не менее $\bar{n}$ требований $j\in\{V_1,V_2,\ldots,V_{2i-1},V_{2i},\ldots,V_{2\bar{n}-1},V_{2\bar{n}}\}$ запаздывает. В доказательстве рассматриваются только расписания вида $\pi=(S,F)=(SPT,LPT)$. Исследуются свойства "стабильных" расписаний. Расписание $\pi=(S,F)$ стабильно, если для каждого незапаздывающего требования $j\in S$ при расписании π выполняется $d_j\geqslant\sum\limits_{i\in S}p_i$, где S – множество незапаздывающих требований. Очевидно, что все оптимальные расписания являются стабильными. Выясним *максимальное количество* k незапаздывающих требований из множества $\bar{V}=\{V_1,V_2,\ldots,V_{2i-1},V_{2i},\ldots,V_{2\bar{n}-1},V_{2\bar{n}}\}$ при стабильном расписании. Наибольшее множество S (с максимальным количеством элементов) содержит требования $V_j,V_{j+1},\ldots,V_{2\bar{n}}$. Тогда $P_S=\sum\limits_{i=j}^{2\bar{n}}p_{V_i}>(2\bar{n}-j+1)a_{2\bar{n}}$. Продолжая эту ветку доказательства, получаем $k\leqslant\left\lceil\dfrac{2}{3}\bar{n}\right\rceil$.

Можно показать, что при всех оптимальных расписаниях все требования W_j, $j=1,\ldots,n+1$, запаздывают и обслуживаются в конце расписания. Это доказывается от противного.

Для оптимального расписания также должно выполняться следующее свойство.

Пусть i – последнее запаздывающее требование из множества $\{V_1, V_2, \ldots, V_{2j-1}, V_{2j}, \ldots, V_{2\bar{n}-1}, V_{2\bar{n}}\}$ при оптимальном расписании $\pi^* = (\pi_1, i, \pi_2)$. Предположим, что при расписании π^* ровно k требований не запаздывает. Тогда перед требованием i обслуживается $2\bar{n} - k - 1$ запаздывающих требований. Поэтому $i \geqslant 2\bar{n} - k$, так как все запаздывающие требования обслуживаются в порядке LPT. Тогда должно выполняться $F(\pi^*) > F(\pi = (i, \pi_1, \pi_2))$, так как расписание π^* оптимальное, т.е. должны иметь

$$C_i(\pi^*) - d_i \geqslant (2\bar{n} - k - 1)p_i \Rightarrow$$
$$\Rightarrow (a_1 + a_3 + \cdots + a_{2\bar{n}-1}) + (a_2 + a_4 + \cdots + a_{2\bar{n}}) - d_i \geqslant (2\bar{n} - k - 1)a_i.$$

Поэтому выполняется $k > \dfrac{2}{3}\bar{n} - 1$.

Если обобщить сказанное выше, то для каждого оптимального расписания количество незапаздывающих требований равно $\dfrac{2}{3}\bar{n} - 1 < k \leqslant \left\lceil \dfrac{2}{3}\bar{n} \right\rceil$ и поэтому $k = \left\lceil \dfrac{2}{3}\bar{n} \right\rceil$ или $\left\lfloor \dfrac{2}{3}\bar{n} \right\rfloor$. Теперь необходимо выбрать k требований и обслужить их в начале расписания. Далее исследуется минимальный номер i требования V_i, которое может не запаздывать в оптимальном расписании $\left(d_i \geqslant \sum\limits_{l \in S} p_l\right)$.

Для всех возможных ситуаций $\left(k = \left\lceil \dfrac{2}{3}\bar{n} \right\rceil \text{ или } \left\lfloor \dfrac{2}{3}\bar{n} \right\rfloor,\ i \text{ четное или нечетное}\right)$ будут получены похожие результаты. Например, для $k = \left\lfloor \dfrac{2}{3}\bar{n} \right\rfloor$ и нечетного i выполняется $i \geqslant \left\lfloor \dfrac{4}{3}\bar{n} \right\rfloor - 3$. Поэтому необходимо выбрать $\left\lfloor \dfrac{2}{3}\bar{n} \right\rfloor$ требований из $2\bar{n} - \left\lfloor \dfrac{4}{3}\bar{n} \right\rfloor + 3$ требований с минимальной продолжительностью из множества $\{V_1, V_2, \ldots, V_{2i-1}, V_{2i}, \ldots, V_{2\bar{n}-1}, V_{2\bar{n}}\}$. Т.е. необходимо рассмотреть порядка $\bar{n}^4$ комбинаций. Данные комбинации (и соответствующие расписания) можно рассмотреть за $O(n^5)$ операций.

3.7. Точный алгоритм решения частного случая задачи $d_1 + p_1 \leqslant d_2 + p_2 \leqslant \ldots \leqslant d_n + p_n,\ p_1 < p_2 < \cdots < p_n$

Без ограничения общности пусть $d_i < \sum\limits_{j=1}^{n} p_j$ для всех $i \in N$. Следующий алгоритм строит оптимальное расписание для данного частного случая.

Алгоритм 3.

0. $P := \sum\limits_{j=1}^{n} p_j,\ \bar{N} := N,\ S := \emptyset,\ \pi = (),\ \Pi := \emptyset$.

1. WHILE *для каждого требования $j \in \bar{N}$, существует требование $i \in \bar{N} \setminus \{j\}$ такое, что $d_i < P - p_j$,* DO

   1.1. Выбираем требование $j^* \in \bar{N}$ с минимальной продолжительностью обслуживания.

   1.2. $\pi = (j^*, \pi),\ P := P - p_{j^*},\ \bar{N} := \bar{N} \setminus \{j^*\}$.

   1.3. FOR *каждого требования i, для которого $d_i \geqslant P$,* DO $\bar{N} := \bar{N} \setminus \{i\},\ \bar{S} := \bar{S} \bigcup \{i\}$.

2. WHILE *существует требование $j \in \bar{N}$ и существует требование $i \in \bar{N} \setminus \{j\}$ такое, что $d_i < P - p_j$* DO

2.1. Рассматриваем все расписания $\pi' = (\pi_1, l, \pi)$, где $\pi_1 = (\bar{N} \setminus \{l\}) \bigcup S$, а требование l такое, что $d_i \geqslant P - p_l$ для всех $i \in \bar{N} \setminus \{l\}$. Лучшее рассмотренное расписание π' помещаем в список П.

2.2. Выбираем требование $j^* \in \bar{N}$ с минимальной продолжительностью обслуживания.

2.3. Рассматриваем все расписания $\pi'' = (\pi_1, l, j^*, \pi)$, где $\pi_1 = (\bar{N} \setminus \{l, j^*\}) \bigcup S$, $d_i \geqslant P - p_l - p_{j^*}$ для всех $i \in \bar{N} \setminus \{l, j^*\}$. Лучшее рассмотренное расписание π'' помещаем в список П.

2.4. В списке П оставляем только лучшее расписание (с наибольшим значением целевой функции).

2.5. $\pi = (j^*, \pi)$, $P := P - p_{j^*}$, $\bar{N} := \bar{N} \setminus \{j^*\}$.

2.6. FOR *каждого требования* i, *для которого* $d_i \geqslant P$, DO $\bar{N} := \bar{N} \setminus \{i\}$, $\bar{S} := \bar{S} \bigcup \{i\}$.

3. Теперь дополнительно можно выбрать только одно запаздывающее требование. Выберем требование $j \in \bar{N}$ с минимальным директивным сроком d_j и построим расписание $\pi = (\pi_1, j, \pi)$, $\pi_1 = (\bar{N} \setminus \{j\}) \bigcup S$. Сравним полученное расписание π с расписанием П и выберем лучшее.

Лемма 22. Алгоритм 3 строит оптимальное расписание для данного частного случая за $O(n^2)$ операций.

Доказательство. Известно, что если $p_i < p_j$ и $d_i \leqslant d_j$, тогда существует оптимальное расписание π, где $(j \to i)_\pi$ (см. правила Эммонса). Если существует расписание $\pi = (\pi_1, i, \pi_2, \alpha, j, \pi_3)$, в котором $p_i < p_j$, $d_i > d_j$, требование i не запаздывает, требования α, j запаздывают, тогда для расписания $\pi' = (\pi_1, j, \pi_2, \alpha, i, \pi_3)$ выполняется $F(\pi') - F(\pi) \geqslant 0$, так как $T_\alpha(\pi') - T_\alpha(\pi) = p_j - p_i$, $(T_j(\pi) - T_j(\pi')) - (T_i(\pi') - T_i(\pi)) \leqslant d_i - d_j$, и $d_i + p_i \leqslant d_j + p_j \Rightarrow d_i - d_j \leqslant p_j - p_i$. Этот факт доказывает оптимальность шагов *1, 2.3. и 2.6.* алгоритма. Очевидно, что трудоемкость алгоритма 3 $O(n^2)$ операций.

3.8. Точный алгоритм решения задачи $1||\max \sum T_j$.

Алгоритм основан на лемме 8, т.е. на том факте, что существует оптимальное расписание вида $\pi = (S, F) = (SPT, LPT)$, где все требования $j \in F$ запаздывают, а все требования $i \in S$ не запаздывают.

Алгоритм 4.

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

2. $\pi_1(t) := (1)$, $F_1(t) := \max\{0, p_1 + t - d_1\}$ для всех целочисленных $t \in Z$, где $t \in \left[0, \sum\limits_{j=2}^{n} p_j\right]$.

3. FOR $l := 2$ TO n DO

FOR $t := 0$ TO $\sum\limits_{j=l+1}^{n} p_j$ $(t \in Z)$ DO

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

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

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

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

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

4. Получено оптимальное расписание $\pi_n(0)$ и соответствующее оптимальное значение целевой функции $F_n(0)$.

Теорема 1. Алгоритм 4 строит оптимальное расписание за $O(n\sum p_j)$ операций.

Доказательство. Доказательство от противного. Допустим, что существует оптимальное расписание $\pi^* = (SPT, LPT)$, для которого $F(\pi^*) > F(\pi_n(0)) = F_n(0)$. Пусть $\pi' := \pi^*$. Для каждого $l = 1, \ldots, n$, 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 $\bar{\pi}_l \neq \pi_l \left(t = \sum\limits_{i \in \pi_\alpha} p_i\right)$ (для определения см. последнюю строку шага 3 алгоритма), тогда $\pi' := \left(\pi_\alpha, \pi_l\left(\sum\limits_{i \in \pi_\alpha} p_i\right), \pi_\beta\right)$. Очевидно, что $F((\pi_\alpha, \bar{\pi}_l, \pi_\beta)) \leqslant F\left(\left(\pi_\alpha, \pi_l\left(\sum\limits_{i \in \pi_\alpha} p_i\right), \pi_\beta\right)\right)$. Аналогичную операцию проделываем для каждого $l = 1, 2, \ldots, n$. В конце получаем $F(\pi^*) \leqslant F(\pi') \leqslant F_n(0)$. Поэтому расписание $\pi_n(0)$ так же оптимально.

Очевидно, что трудоемкость алгоритма составляет $O(n\sum p_j)$ операций, так как на шаге 3 алгоритма выполняются $n-1$ итераций и просматриваются все целочисленные точки из интервала, ограниченного интервалом $\left[0, \sum\limits_{j=1}^{n} p_j\right]$.

Для практической реализации алгоритма можно использовать идею, описанную в [13]. Вычислительный эксперимент для задачи "Разбиение" показал, что с помощью данной идеи существенную часть примеров задачи "Разбиение" можно решать за полиномиальное время. Ожидается, что данное свойство будет выполняться и для данной задачи.

3.9. Альтернативный точный алгоритм решения задачи $1||\max\sum T_j$.

Перенумеруем требования согласно правилу $d_1 \leqslant d_2 \leqslant \ldots \leqslant d_n$, если $d_i = d_{i+1}$, то $p_i \leqslant p_{i+1}$ (EDD порядок). Пусть j^* – требование с максимальной продолжительностью $j^* := \operatorname{argmax}\left\{d_i : p_i = \max\limits_{k \in N} p_k\right\}$. Обозначим через jf первое запаздывающее требование при оптимальном расписании $\pi = (S, F)$. Будем обозначать $P(A) = \sum\limits_{i \in A} p_i$.

В соответствии с леммами 9 и 12 для всех незапаздывающих требований $i \in S$ выполняется $d_i \geqslant \max\left\{d_{jf}, \sum\limits_{k \in S} p_k\right\}$, где $\sum\limits_{k \in S} p_k = S_{jf}(\pi)$ и $S_{jf}(\pi)$ – время начала обслуживания требования jf при расписании π.

Известно, что требование j может быть первым запаздывающим требованием, только если не существует требования k, что $k < j$, $d_k < d_j$, $p_k > p_j$ (см. лемму 14). Пусть $J = \{j_1, j_2, \ldots, j^*\}$, где $N = \{1, 2, \ldots, j_1 - 1, j_1, j_1 + 1, \ldots, j_2, \ldots, j^*, \ldots, n\}$ – множество требований, которые могут быть первым запаздывающим требованием при оптимальном расписании. Если j_k, $j_l \in J$, $j_k < j_l$, тогда $S_{j_k} < d_{j_l}$ в оптимальном расписании, где j_k – первое запаздывающее требование.

Для каждого требования $j_k \in J$, можно рассмотреть все ситуации $S_{j_k} \in (d_\alpha, d_{\alpha+1}]$, $\alpha = j_k, j_k + 1, \ldots, j_{k+1} - 1$ и $S_{j_k} \in (d_{j_k} - p_{j_k}, d_{j_k}]$, где S_{j_k} – время начала обслуживания требования j_k при оптимальном расписании. Очевидно, что существует ситуация $(j_k,\ S_{j_k})$, которая соответствует оптимальному расписанию, где $j_k \in J$ – первое запаздывающее требование, и $S_{j_k} \in (d_\alpha, d_{\alpha+1}]$, $\alpha = j_k, j_k + 1, \ldots, j_{k+1} - 1$ или $S_j \in (d_{j_k} - p_{j_k}, d_{j_k}]$. Каждую из данных ситуаций можно рассматривать отдельно. Существует не более n таких ситуаций.

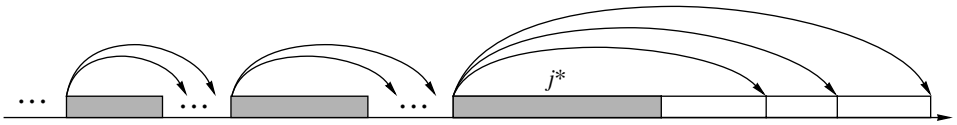

Рис. 1. EDD порядок. Выбор ситуации в оптимальном расписании.

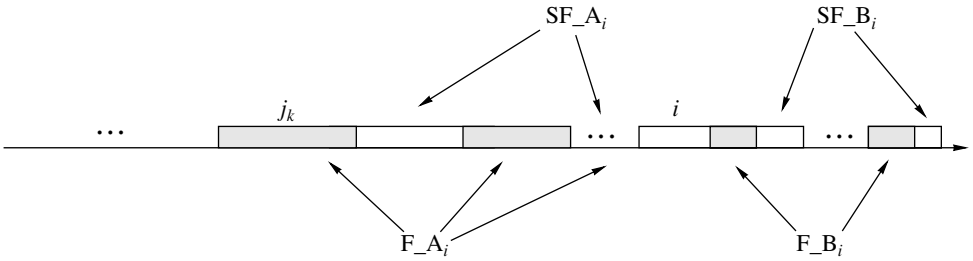

Рис. 2. Множества требований в функции Ветвление.

Если принять, что $j_k \in J$ – первое запаздывающее требование и $S_{j_k} \in (d_\alpha, d_{\alpha+1}]$ в оптимальном расписании, то все требования $(1, \ldots, \alpha)$ также запаздывают. Без потери общности можно принять $d_{j_k} := d_{\alpha+1}$ и рассматривать новый пример с модифицированным директивным сроком d_{j_k} (см. рис. 1 с EDD расписанием).

Предлагается следующий точный алгоритм.

Алгоритм 5.

1. Вычислим $j^* := \operatorname{argmax}\left\{d_i : p_i = \max\limits_{k \in N} p_k\right\}$ и множество $J = \{j_1, j_2, \ldots, j^*\}$ требований, которые могут быть первыми запаздывающими требованиями при оптимальном расписании.
2. FOR каждого требования $j_k \in J$ DO:

   Рассматриваем отдельно каждый интервал $(d_\alpha, d_{\alpha+1}]$, $\alpha = j_k, j_k + 1, \ldots$ $\ldots, j_{k+1} - 1$ и интервал $(d_{j_k} - p_{j_k}, d_{j_k}]$. Пусть $S_{j_k} \in (d_\alpha, d_{\alpha+1}]$. Тогда примем $F := \{i \in N, d_i < d_{\alpha+1}\} \bigcup \{j_k\}$, $S := \{i \in N, p_i > p_{j_k}\}$ и $SF := N \setminus F \setminus S$. Для данной ситуации оптимальным является расписание $\pi =$ $=$ Ветвление(F, S, SF, j_k, α). Сравним полученное расписание с текущим лучшим расписанием.

Ветвление(F, S, SF, j_k, α)

1. Для каждого требования $i \in F \bigcup SF$ определим множества требований $F\_A_i =$ $= \{j \in F,\ p_j \geqslant p_i\}$, $F\_B_i = \{j \in F,\ p_j < p_i\}$, $SF\_A_i = \{j \in SF,\ p_j \geqslant p_i\}$ и $SF\_B_i = \{j \in SF,\ p_j < p_i\}$ (см. рис. 2).
2. **Правило сокращения 1.** Если для требования $j \in F$ существует требование $k \in SF\_B_j$ такое, что $d_k - d_j < |F\_A_j|(p_j - p_k)$, где $|F\_A_j|$ – количество элементов (требований) в множестве F_A_j, тогда $F := F \bigcup \{k\}$, $SF := SF \setminus \{k\}$. Доказательство очевидно. Если в оптимальном расписании π требование k не запаздывает, а требование j запаздывает, тогда можно поменять местами эти требования в расписании и для нового расписания π' получить $F(\pi') > F(\pi)$.
3. **Правило сокращения 2.** Если для требования $j \in F$ существует требование $k \in$ $\in SF\_B_j$ такое, что $d_k \geqslant d_j$, тогда $F := F \bigcup \{k\}$, $SF := SF \setminus \{k\}$.

4. **Правило сокращения 3.** Известно, что для данной ситуации (F, S, SF, j_k, α) выполняется $S_{j_k} \leqslant d_{\alpha+1}$. Если для всех $k \in SF\_A_i \bigcup F\_A_i$, где $d_{\alpha+1} + P(SF\_A_k) + P(F\_A_k) + p_k \geqslant \sum_{j=1}^{n} p_j - P(SF\_B_i) - P(SF\_B_i)$, выполняется $d_i - d_k < |F\_A_k|(p_k - p_i)$, тогда $F := F \bigcup \{i\},\ SF := SF \setminus \{i\}$.
5. **Правило сокращения 4.** Если для всех $k \in SF\_B_i \bigcup F\_B_i$, где $d_{\alpha+1} + P(SF\_A_i) + P(F\_A_i) \geqslant \sum_{j=1}^{n} p_j - P(SF\_B_k) - P(SF\_B_k) - p_k$, выполняется $d_k - d_i > |F\_A_i \bigcup SF\_A_i|(p_i - p_k)$, тогда $F := F \bigcup \{i\},\ SF := SF \setminus \{i\}$.
6. Строим расписание вида $\pi = (SPT, LPT)$. Все требования из множества S обслуживаются в начале расписания в порядке SPT. Все требования из множеств F и SF обслуживаются в конце расписания в порядке LPT. Если для всех требований из F и SF все требования при данном расписании запаздывают, тогда возвращается π.
7. Пусть $l \in F \bigcup SF$ – последнее незапаздывающее требование при расписании π. Очевидно, что не существует оптимального расписания, при котором все требования из множества $l \bigcup SF\_B_l \bigcup F\_B_l$ запаздывают. Поэтому необходимо выбрать одно требование i из множества SF_B_l (или из множества $SF\_B_l \bigcup \{l\}$, если $l \in SF$) и поместить это требование во множество S.
8. Пусть $i \in SF\_B_l$ (или $i = l$, если $l \in SF$) – требование с максимальной продолжительностью из множества SF_B_l (или $i = l$, если $l \in SF$).
9. Вычисляем

$$\pi_1 = \text{Ветвление}\left(F \bigcup \{i\}, S, SF \setminus \{i\}, j_k, \alpha\right)$$

и

$$\pi_2 = \text{Ветвление}\left(F, S \bigcup \{i\}, SF \setminus \{i\}, j_k, \alpha\right)$$

и возвращаем лучшее расписание из π_1 и π_2.

4. Заключение

В работе предложен полиномиальный точный алгоритм решения задачи максимизации количества запаздывающих требований $1||\max \sum U_j$, когда простои в обслуживании требований запрещены. Предложены два точных алгоритма решения задачи максимизация суммарного запаздывания $1||\max \sum T_j$. Показано, что известные NP-трудные частные случаи задачи минимизации суммарного запаздывания, являются полиномиально разрешимыми для задачи максимизации.

Через три месяца после принятия статьи в печать нами было доказано, что задача $1\|\max \sum T_j$ полиномиально разрешима за время $O(n^2)$, а задачи $1|r_j|\max \sum T_j$ и $1\|\max \sum w_j T_j$ NP-трудны в обычном смысле [14].

СПИСОК ЛИТЕРАТУРЫ

1. *Moore J.M.* An n job, one machine sequencing algorithm for minimizing the number of late jobs // Management Sci. 1968. V. 15. No 1. P. 102–109.
2. *Du J., Leung J. Y.-T.* Minimizing Total Tardiness on One Processor is NP-hard // Math. Oper. Res. 1990. V. 15. P. 483–495.
3. *Гафаров Е.Р., Лазарев А.А.* Доказательство NP-трудности частного случая задачи минимизация суммарного запаздывания для одного прибора // Изв. РАН: Теория и системы управления. 2006. № 3. Р. 120–128.

4. *Lawler E.L.* A Pseudopolynomial Algorithm for Sequencing Jobs to Minimize Total Tardiness // Ann. Discrete Math. 1977. V. 1. C. 331–342.
5. *Szwarc W., Della Croce F., Grosso A.* Solution of the Single Machine Total Tardiness Problem // J. Scheduling. 1999. V. 2. P. 55–71.
6. *Potts C.N., Van Wassenhove L.N.* A Decomposition Algorithm for the Single Machine Total Tardiness Problem // Oper. Res. Lett. 1982 V. 1. P. 363–377.
7. *Лазарев А.А., Гафаров Е.Р.* Теория расписаний. Минимизация суммарного запаздывания для одного прибора // Науч. изд. М.: Вычислит. центр им. А.А. Дороницына РАН. 2006. 134 с.
8. *Lazarev A.* Dual of the maximum cost minimization problem // J. Math. Sci. Springer. 1989. V. 44. No 5. P. 642–644.
9. *Huang, R.H., Yang, C.L.* Single-Machine Scheduling to Minimize the Number of Early Jobs // IEEE Int. Conf. Indust. Engin. Engin. Management. 2007. art. no. 4419333. P. 955–957.
10. *Lazarev A.A., Werner F.* Algorithms for Special Cases of the Single Machine Total Tardiness Problem and an Application to the Even-Odd Partition Problem // Mathematical and Computer Modelling. 2009. V. 49. No 9–10. P. 2061–2072.
11. *Лазарев А.А., Кварацхелия А.Г., Гафаров Е.Р.* Алгоритмы решения NP-трудной проблемы минимизации суммарного запаздывания для одного прибора // Докл. АН. 2007. V. 412. № 6. С. 739–742.
12. *Gafarov E.R., Lazarev A.A., Werner F.* Algorithms for Maximizing the Number of Tardy Jobs or Total Tardiness on a Single Machine // Preprint 38/09. FMA, Otto-von-Guericke-Universität Magdeburg. 2009.
13. *Lazarev A.A., Werner F.* A Graphical Realization of the Dynamic Programming Method for Solving NP-Hard Combinatorial Problems // Comput. Math. Appl. 2009. V. 58. No 4. P. 619–631.
14. *Gafarov E.R., Lazarev A.A., Werner F.* A Modification of Dynamic Programming Algorithms to Reduce the Running Time or/and Complexity // Preprint 20/10. FMA, Otto-von-Guericke-Universität Magdeburg. 2010.

Статья представлена к публикации членом редколлегии П.Ю. Чеботаревым.

Поступила в редакцию 12.01.2010