Алгоритмы решения задач теории расписаний на однопутной железной дороге[1]

Е.Р. Гафаров

Институт проблем управления им. В.А. Трапезникова РАН, г. Москва, email: axel73@mail.ru

А.А. Лазарев

Институт проблем управления им. В.А. Трапезникова РАН, г. Москва, Московский государственный университет им. М.В. Ломоносова, г. Москва, Национальный Исследовательский Университет Высшая Школа Экономики, г. Москва, email: jobmath@mail.ru

Задача поиска оптимального расписания движения поездов между двумя станциями, соединенными однопутной железной дорогой, (Single Track Railway Scheduling Problem with two stations, STRSP2) формулируется следующим образом.

Пусть однопутная железная дорога, соединяющая 2 станции, разделена на Q сегментов $1, 2, \dots, Q$, и заданы множества N'_1 и N'_2 поездов, $N'_1 \bigcap N'_2 = \emptyset$. Поезда из множества N'_1 следуют со станции 1 на станцию 2, а поезда из N'_2 движутся в противоположном направлении. Множество $N' = N'_1 \bigcup N'_2$ содержит n' поездов, $|N'_1| = n'_1$, $N'_2 = n'_2$, $n'_1 + n'_2 = n'$. Поезда из множества N'_1 проходят сегменты в порядке $1 \to 2 \to \dots \to Q$, а поезда из N'_2 в порядке $Q \to Q-1 \to \dots \to 1$. На одном сегменте дороги не может находиться одновременно более одного поезда. Если поезд $j' \in N'_1$ находится на некотором сегменте пути, то ни один из поездов $i' \in N'_2$ не может начать движение и наоборот. Для каждого сегмента q, $q = 1, 2 \dots, Q$, задано время прохождения p_q, за которое любой поезд $j' \in N'$ его проходит, т.е. все поезда движутся с одинаковой скоростью. Расписание движения поездов задается моментами начала и окончания их движения по рассматриваемому участку. Пусть $S_{j'}(\Pi)$ и $C_{j'}(\Pi)$, $j' \in N'$ — моменты начала и окончания движения поезда j' при расписании Π, т.е. $S_{j'}(\Pi)$ — время выхода поезда j' со станции отправления, а $C_{j'}(\Pi)$ — время прибытия поезда на станцию назначения. Тогда в допустимом расписании выполняются следующие условия:

- $C_{j'} \geq S_{j'} + \sum\limits_{q=1}^{Q} p_q,\ j' \in N'$;

[1]Работа выполнена при финансовой поддержке гранта РФФИ-РЖД 11-08-13121-офи-м-2011-РЖД

- $C_{i'} \leq S_{j'}$ или $C_{j'} \leq S_{i'}$, $\forall i' \in N'_1$, $\forall j' \in N'_2$.

Для каждого поезда $j' \in N'$ заданы директивный срок $d_{j'} \geq 0$, приоритет (вес) $w_{j'} \geq 0$ и время, с которого поезд доступен для отправления $r_{j'} \geq 0$ (самый ранний из возможных моментов отправления $S_{j'} \geq r_{j'}$). Если $C_{j'}(\Pi) > d_{j'}$, то говорят, что поезд j' запаздывает, при этом принимают $U_{j'}(\Pi) = 1$, иначе $U_{j'}(\Pi) = 0$. Если $C_{j'}(\Pi) \leq d_{j'}$, то поезд j' не запаздывает. Обозначим $T_{j'}(\Pi) = \max\{0, C_{j'}(\Pi) - d_{j'}\}$ – запаздывание поезда j' при расписании Π и $C_{max}(\Pi) = \max\limits_{j' \in N'}\{C_{j'}(\Pi)\}$ время окончания всех перевозок при этом расписании. Задачу STRSP2 поиска оптимального расписания π^*, минимизирующего время окончания перевозок C_{max}, обозначим $STRSP2|r_j|C_{max}$ (в соответствии с трехпозиционной системой обозначений $\alpha|\beta|\gamma$ для задач теории расписаний [2], где α обозначает множество ресурсов, β описывает ограничения, γ содержит целевую функцию). Дополнительно будем рассматривать задачу $STRSP2$ с другими целевыми функциями и ограничениями:

- минимизация числа запаздывающих поездов при их одновременном поступлении $STRSP2||\sum U_j$;

- минимизация взвешенного числа запаздывающих поездов при их одновременном поступлении $STRSP2||\sum w_j U_j$;

- минимизация суммарного времени перевозок $STRSP2|r_j|\sum C_j$ при заданных моментах поступления поездов на станции отправления;

- минимизация взвешенного суммарного веремени при одновременном поступлении поездов на станции отправления $STRSP2||\sum w_j C_j$.

Нам не известны публикации по сформулированным выше задачам. Однако стоит отметить, что при $Q = 1$ данные задачи эквиваленты классическим одноприборным задачам теории расписаний [1] и, следовательно, в случае различных скоростей для поездов на сегменте, некоторые из задач оказываются NP-трудными [1]. Заметим также, что данные задачи могут быть легко сформулированы в терминах многоприборных задач теории расписаний с Q приборами [1].

Далее представлена полиномиальное сведение задач $STRSP2$ к специальному случаю одноприборной задачи с одинаковым временем обслуживания всех требований и временем переналадки, а также предлагаются полиномиальные алгоритмы минимизации указанных выше целевых фукнций для полученной одноприборной задачи.

Сведение STRSP2 к одноприборной задаче

Введем обозначения: $p_{max} = \max\limits_{q=1,2,\ldots,Q}\{p_q\}$ и $P = \sum\limits_{q=1}^{Q} p_q$.

Лемма 1 *Пусть для поезда $j' \in N_1'$ время прибытия на станцию назначения определяется как $C_{j'} = S_{j'} + P$, а поезд $i' \in N_1'$ — следующий за ним поезд. Тогда существует допустимое расписание, при котором $S_{i'} = \max\{r_{i'}, S_{j'} + p_{max}\}$, $C_{i'} = S_{i'} + P$, т.е. поезд i' начинает движение со станции отправления в момент $\max\{r_{i'}, S_{j'} + p_{max}\}$ и движется без остановок.*

Доказательство. Пусть $S_{j'}^q$, $S_{i'}^q, q = 1, 2, \ldots, Q,$ — моменты начала прохождения отрезка q поездами j' и i' соответственно. Чтобы доказать допустимость рассматриваемого расписания, достаточно показать, что $S_{i'}^q \geq S_{j'}^q + p_q, q = 1, 2, \ldots, Q$, т.е. поезд i' начинает движение по сегменту q, когда поезд j' уже прошел его. Имеем $S_{j'}^q = S_{j'} + \sum\limits_{l=1}^{q-1} p_l$ и $S_{i'}^q = S_{j'} + \sum\limits_{l=1}^{q-1} p_l = \max\{r_{i'}, S_{j'} + p_{max}\} + \sum\limits_{l=1}^{q-1} p_l \geq S_{j'}^q + p_q$, т.е. лемма верна. $\square$

Лемма 1 определяет периодичность отправления поездов с одной станции, если в расписании они следуют друг за другом. Необходимо отметить, что $\max\{r_{i'}, S_{j'} + p_{max}\}$ — самый ранний из возможных моментов отправления для поезда i', т.к. для участка пути q величина $p_q = p_{max}$. Получаем $S_{i'}^q = S_{j'}^q + p_q$ и, следовательно, $|C_{j'} - C_{i'}| \geq p_{max}$ для любых поездов j', i', принадлежащих одному из множеств N_1' или N_2'.

Из леммы можно сделать также следующий вывод. Для указанных выше целевых функций существует оптимальное расписание, при котором поезда движутся без остановок, т.е. начав движение во время $S_{j'}$ поезд j' прибывает на станцию назначения во время $C_{j'} = S_{j'} + P$. Далее мы будем рассматривать только расписания данного вида.

Лемма 2 *Для любых двух поездов j' и i', принадлежащих одному из множеств N_1' или N_2' выполняется $|S_{j'} - S_{i'}| \geq p_{max}$ и $|C_{j'} - C_{i'}| \geq p_{max}$.*

Пусть последовательность поездов $(j_1', j_2', \ldots, j_{n'}')$ задает очередность движения поездов между станциями. Очевидно, что допустимому расписанию соответствует одна и только одна последовательность поездов. Таким образом, оптимальное расписание соответствует *оптимальной последовательности* поездов. Для заданной последовательности $(j'_1, j'_2, \ldots, j'_{n'})$ расписание можно построить следующим образом:

$$\begin{cases} S_{j'_1} = r_{j'_1}, & C_{j'_1} = S_{j'_1} + P, \\ S_{j'_k} = \max\{r_{j'_k}, S_{j'_{k-1}} + p_{max}\}, & C_{j'_k} = S_{j'_k} + P, \quad k = 2, \ldots, n', (*) \\ S_{j'_k} = \max\{r_{j'_k}, S_{j'_{k-1}} + P\}, & C_{j'_k} = S_{j'_k} + P, \quad k = 2, \ldots, n', (**). \end{cases} \tag{1}$$

$(*)$ выполняется, если оба поезда j'_k и j'_{k-1} принадлежат одному множеству N'_1 или N'_2, иначе выполняется $(**)$.

Согласно лемме 1 построенное расписание будет допустимым. Более того, для представленных выше целевых функций, монотонных по времени окончания движения, в соответствии с леммой 2 алгоритм (1) по заданной оптимальной последовательности поездов построит оптимальное расписание.

На основе данных свойств далее предлагается следующее сведение исходной задачи к одноприборной задаче теории расписаний.

Одноприборная задача

Задано множество $N = N_1 \bigcup N_2,\ N_1 \bigcap N_2 = \emptyset$, содержащее n требований, которые должны быть обслужены на одном приборе. Прерывание обслуживания требований не допускается. Одновременно прибором может обслуживаться не более одного требования. Продолжительность обслуживания требования равно p, $\forall j \in N$. Для каждого требования $j \in N$ заданы директивный срок $d_j \geq 0$, вес (важность) $w_j \geq 0$, и время поступления требования на обслуживание $r_j \geq 0$. Допустимое решение задается перестановкой $\pi = (j_1, j_2, \ldots, j_n)$ требований из множества N, из которой соответствующее расписание может быть получено назначением каждому требованию наиболее раннего из возможных моментов начала обслуживания. Пусть $S_{j_k}(\pi),\ C_{j_k}(\pi) = S_{j_k}(\pi) + p$ — моменты начала и окончания обслуживания требования j_k при расписании, полученном из перестановки π.

Если $j_k \in N_1$ и $j_{k+1} \in N_2$, то между обслуживанием требований необходимо выполнить переналадку прибора продолжительностью $st = st_1$. Если $j_k \in N_2$ и $j_{k+1} \in N_1$, то между обслуживанием требований необходимо выполнить переналадку прибора продолжительностью $st = st_2$. Между обслуживанием требований из одного множества переналадка не требуется, т.е. $st = 0$. В допустимом расписании выполняется условие $S_{j_{k+1}} = \max\{r_{j_{k+1}}, C_{j_k} + st\}$. Будем рассматривать те же целевые функции, что и для задачи $STRSP2$. Если $C_j(\pi) > d_j$, то требование j

запаздываем, и полагают $U_j(\pi) = 1$, иначе $U_j(\pi) = 0$. Если $C_j(\pi) \leq d_j$, то требование j не запаздывает. Пусть $T_j(\pi) = \max\{0, C_j(\pi) - d_j\}$ — запаздывание работы j, а $C_{max}(\pi) = \max_{j \in N}\{C_j(\pi)\}$ — время окончания всех работ.

Обозначим задачу минимизации времени окончания обслуживания всех требований для одного прибора с одинаковым временем обслуживания, заданными временами поступления работ и заданным временем переналадки через $1|setup-times, N_1, N_2, p_j = p, r_j|C_{max}$ в соответствии с общепринятой классификацией.

Задачи $STRSP2|-|-$ для перечисленных выше целевых функций могут быть сведены к задачам $1|setup-times, N_1, N_2, p_j = p, -|-$ следующим образом. Подмножество поездов N_1' соответствует подмножеству требований N_1, $|N_1| = |N_1'|$, а подмножество поездов N_2' — подмножеству требований N_2, $|N_2| = |N_2'|$. Пусть q, $q \in \{1, 2, \ldots, Q\}$ — индекс участка, для которого $p_q = p_{max}$. Обозначим $TAIL_{left} = \sum_{l=1}^{q-1} p_l$, $TAIL_{right} = \sum_{l=q+1}^{Q} p_l$. Положим $p = p_{max}$, $st_1 = 2 \cdot TAIL_{right}$, $st_2 = 2 \cdot TAIL_{left}$, если $j \in N_1$, то время поступления работы $r_j = r_{j'} + TAIL_{left}$, иначе $r_j = r_{j'} + TAIL_{right}$. Если $j \in N_1$, то $d_j = d_{j'} - TAIL_{right}$, в противном случае $d_j = d_{j'} - TAIL_{left}$. Веса остаются теми же.

Рассмотрим последовательность обслуживания требований $(j_1, j_2, \ldots, j_n)$, соответствующую последовательности движения поездов $(j_1', j_2', \ldots, j_{n'}')$ между станциями, где требование $j_k, k = 1, 2, \ldots, n$, соответствует поезду j_k', а также расписания, полученные из данных последовательностей, в которых каждое/ый требование/поезд обслуживается/начинает движение как можно раньше (для требований) или в соответствии с алгоритмом (1) (для поездов).Тогда для требования j и поезда j' можно составить следующую таблицу соответствий:

Таблица 1: Соответствие параметров.

требование/поезд	время поступления	директ. срок	время старта	время окончания
$j \in N_1$	$r_j = r_{j'} + TAIL_{left}$	$d_j = d_{j'} - TAIL_{right}$	$S_{j'} + TAIL_{left}$	$C_{j'} - TAIL_{right}$
$j' \in N_1'$	$r_{j'}$	$d_{j'}$	$S_{j'}$	$C_{j'}$
$j \in N_2$	$r_j = r_{j'} + TAIL_{right}$	$d_j = d_{j'} - TAIL_{left}$	$S_{j'} + TAIL_{right}$	$C_{j'} - TAIL_{left}$
$j' \in N_2'$	$r_{j'}$	$d_{j'}$	$S_{j'}$	$C_{j'}$

В следующей таблице указаны соответствия целевых функций задач.
Таблица 2: Соответствие значений целевых функций.

Целевая функция задачи $STRSP2\|-\|-$	Целевая функция задачи $1\|setup-times, N_1, N_2, p_j = p, -\|-$
$\sum w_{j'}U_{j'}$	$\sum w_{j'}U_{j'}$
$\sum T_{j'}$	$\sum T_{j'}$
$\sum w_{j'}C_{j'}$	$\sum w_{j'}C_{j'} + \sum_{j' \in N'_1} w_{j'} \cdot TAIL_{right} + \sum_{j' \in N'_2} w_{j'} \cdot TAIL_{left}$

Таким образом, можно заключить, что для указанных в таблице целевых функций оптимальная последовательность обслуживания требований $(j_1, j_2, \ldots, j_n)$ соответствует оптимальной последовательности движения поездов $(j'_1, j'_2, \ldots, j'_{n'})$, т.е. задачи $STRSP2|-|-$ можно свести к соответствующим задачам $1|setup - times, N_1, N_2, p_j = p, -|-$ за полиномиальное время. В полученных задачах для одного прибора обслуживание всех требований $j \in N_1$ начинается не раньше времени $r = TAIL_{left}$, а для требований $j \in N_2$ – не раньше момента времени $r = TAIL_{right}$. Предлагаемые далее алгоритмы решения предполагают, что все моменты поступления r равны 0, однако данные алгоритмы могут быть легко преобразованы для решения исходных задач с временами поступления не равными 0.

Можно представить аналогичное сведение задачи $STRSP2|r_j|C_{max}$ к задаче $1|setup-times, N_1, N_2, p_j = p, r_j|C_{max}$, однако в этом случае не будет строгого соответствия оптимальных значений функции C_{max}, т.е. оптимальная последовательность выполнения требований $(j_1, j_2, \ldots, j_n)$ может соответствовать неоптимальной последовательности движения поездов $(j'_1, j'_2, \ldots, j'_{n'})$. Тем не менее, алгоритм решения задачи $1|setup-times, N_1, N_2, p_j = p, r_j|C_{max}$ может быть модифицирован для решения задачи $STRSP2|r_j|C_{max}$.

Таким образом, далее представлены алгоритмы решения следующих задач.

- $1|setup - times, N_1, N_2, p_j = p, r_j|C_{max}$;
- $1|setup - times, N_1, N_2, p_j = p, r_j \sum C_j$;
- $1|setup - times, N_1, N_2, p_j = p| \sum w_j C_j$;
- $1|setup - times, N_1, N_2, p_j = p| \sum T_j$;
- $1|setup - times, N_1, N_2, p_j = p| \sum U_j$;

- $1|setup-times, N_1, N_2, p_j = p|\sum w_j U_j$.

Обзор по одноприборным задачам с переналадками представлен, например, в [3]. Некоторые результаты для одноприборных задач с равными продолжительностями обслуживания требований описаны в [4, 5].

Определение 1. Назовем расписания для задач $1|setup-times, N_1, N_2, p_j = p, -|-$ *"сдвинутыми влево"*, если обслуживание каждого требования начинается в самый ранний из возможных моментов времени. Очевидно, что для любой из упомянутых выше задач существуют оптимальные расписания, которые являются "сдвинутыми влево".

Определение 2. Определим множество Θ следующим образом: $\Theta = \{t|t = r_j + x_1 \cdot p + x_2 \cdot st_1 + x_3 \cdot st_2,\ j \in \{1, 2, \ldots, n\},\ x_1, x_2, x_3 \in \{0, 1, 2, \ldots, n\},\ x_2 + x_3 \leq x_1\}$.

Заметим, что множество Θ содержит не более $O(n^4)$ элементов.

Лемма 3 *Во всех "сдвинутых влево" расписаниях моменты начала обслуживания требований принадлежат множеству* Θ.

Доказательство. Пусть в допустимом "сдвинутом влево" расписании Π требование $j_k,\ 1 < k < n$, является первым требованием, для которого $S_{j_k} \notin \Theta$, т.е для предшествующего ему требования j_{k-1} выполнено $S_{j_{k-1}} \in \Theta$. Самый ранний из возможных моментов S начала обслуживания требования j_k определяется следующим образом:

$$\left\{ \begin{array}{ll} S = \max\{r_{j_k}, S_{j_{k-1}} + p\}, & (*) \\ S = \max\{r_{j_k}, S_{j_{k-1}} + p + st_1\}, & (**) \\ S = \max\{r_{j_k}, S_{j_{k-1}} + p + st_2\}. & (***) \end{array} \right.$$

(*) выполняется, если требования j_k и j_{k-1} принадлежат одному и тому же множеству N_1 или N_2, (**) выполняется, когда $j_k \in N_2$ и $j_{k-1} \in N_1$, а (***) справедливо для случая, когда $j_k \in N_1$ и $j_{k-1} \in N_2$. Очевидно, что $S \in \Theta$. Поскольку $S_{j_k} \notin \Theta$, получаем $S < S_{j_k}$, а значит, расписание Π не является "сдвинутым влево". $\square$

Алгоритмы для задач с упорядоченными подмножествами N_1 и N_2 Далее представлены алгоритмы решения следующих задач:

- $1|setup-times, N_1, N_2, p_j = p, r_j|C_{max}$;

- $1|setup-times, N_1, N_2, p_j = p, r_j \sum C_j$;

- $1|setup-times, N_1, N_2, p_j = p|\sum w_j C_j$;

- $1|setup - times, N_1, N_2, p_j = p|\sum T_j$.

Все алгоритмы основаны на одних и тех же свойствах оптимальных решений и используют одну и ту же процедуру поиска.

Обозначим: $N_1 = \{j_1, j_2, \dots, j_{n_1}\}$ и $N_2 = \{i_1, i_2, \dots, i_{n_2}\}$.

Лемма 4 *Для всех указанных выше задач существует оптимальное расписание, при котором требования выполняются в одной из следующих последовательностей.*

- *для задач $1|setup - times, N_1, N_2, p_j = p, r_j|C_{max}$ и $1|setup - times, N_1, N_2, p_j = p, r_j|\sum C_j$ требования из каждого подмножества N_1 и N_2 обслуживаются в порядке по неубыванию моментов поступления, т.е. $r_{j_1} \le r_{j_2} \le \dots \le r_{j_{n_1}}$ и $r_{i_1} \le r_{i_2} \le \dots \le r_{i_{n_2}}$;*

- *для задач $1|setup - times, N_1, N_2, p_j = p|\sum w_j C_j$ требования из каждого подмножества N_1 и N_2 обслуживаются в порядке по невозрастанию весов, т.е. $w_{j_1} \ge w_{j_2} \ge \dots \ge w_{j_{n_1}}$ и $w_{i_1} \ge w_{i_2} \ge \dots \ge w_{i_{n_2}}$;*

- *для задач $1|setup - times, N_1, N_2, p_j = p|\sum T_j$ требования из каждого подмножества N_1 и N_2 обслуживаются в порядке по неубыванию директивных сроков, т.е. $d_{j_1} \le d_{j_2} \le \dots \le d_{j_{n_1}}$ и $d_{i_1} \le d_{i_2} \le \dots \le d_{i_{n_2}}$.*

Доказательство. Лемма может быть легко доказана следующим образом. Если в оптимальном расписаним Π два требования, принадлежащие одному и тому же подмножеству N_1 или N_2, обслуживаются с нарушением указанного порядка, то в расписании их можно поменять местами без увеличения значения целевой функции. $\square$

Далее представлен алгоритм решения задачи $1|setup - times, N_1, N_2, p_j = p, r_j|C_{max}$, и дается объяснение, как он может быть использован для решения других рассматриваемых задач.

Предположим, что требования в N_1 и N_2 упорядочены в соответствии с леммой 4. В алгоритме одно за другим рассматриваются требования $i_1, i_2, \dots, i_{n_2}$. Для каждого требования $i_k,\ k = 1, 2, \dots, n_2$, необходимо рассмотреть все позиции $l,\ l = 0, 1, 2, \dots, n_1$, где позиция l означает, что требование j_l предшествует требованию i_k в конструируемом расписании, и i_k предшествует требованию j_{l+1}. Если для требования i_k выбрана позиция l, то для требования i_{k+1} могут быть рассмотрены только позиции $l, l+1, \dots, n_1$ (см. лемму 4).

Function $Sequence(k, l, S_{i_{k-1}})$

if $k = n_2 + 1$ **then**
  Назначить обслуживание требований $j_{l+1}, j_{l+2}, \ldots, j_{n_1}$ с момента времени $S_{i_{k-1}} + p + st_1$ соласно алгоритму (1);
  $\sigma := (j_{l+1}, j_{l+2}, \ldots, j_{n_1})$
  Вернуть пару $(C_{j_{n_1}}, \sigma)$;
end if
if $l = n_1$ **then**
  Назначить обслуживание требований $i_k, i_{k+1}, \ldots, i_{n_2}$ с момента времени $S_{i_{k-1}} + p$ соласно алгоритму (1);
  $\sigma := (i_k, i_{k+1}, \ldots, i_{n_2})$
  Вернуть пару $(C_{i_{n_2}}, \sigma)$;
end if
$S := S_{i_{k-1}}$;
$f_{min} := \infty$;
$\sigma_{min} := ()$;
for $pos := l$ to n_1 **do**
  if $pos = l$ **then**
    $S_{i_k} := \max\{r_{i_k}, S + p\}$;
    $S := S + p + st_1$;
    $(f, \sigma) := Sequence(k + 1, l, S_{i_k})$;
  else
    $S_{j_{pos}} := \max\{r_{j_{pos}}, S\}$;
    $S_{i_k} := \max\{r_{i_k}, S_{j_{pos}} + p + st_2\}$;
    $(f, \sigma) := Sequence(k + 1, pos, S_{i_k})$;
    $S := S_{j_{pos}} + p$;
  end if
  if $f_{min} > f$ **then**
    $f_{min} := f$;
    $\sigma_{min} := (j_{l+1}, \ldots, j_{pos}, i_k, \sigma)$
  end if
end for
Вернуть пару (f_{min}, σ_{min});

Алгоритм 2.
$(F, \pi_{opt}) := Sequence(1, 0, -p)$, где π_{opt} — оптимальная последовательность обслуживания требований, $F = C^*_{max}$ — минимальное время окончания обслуживания всех требований.

Оценим трудоемкость алгоритма. Множество требований для которых еще не составлено расписание, являющиеся аргументами рекурсивной процедуры имеют вид

$$\{j_{l+1}, j_{l+2}, \dots, j_{n_1}, i_k, i_{k+1}, \dots, i_{n_2}\},$$

т.е. множества точно определяются парой индексов (k, l). Аргументы $S_{i_{k-1}}$ принадлежат множеству Θ. Следовательно, процедура $Sequence(k, l, S_{i_{k-1}})$ будет вызвана не более $O(n^6)$ раз. Трудоемкость процедуры составляет $O(n)$ операций. Таким образом, трудоемкость алгоритма 2 составляет $O(n^7)$ операций.

В соответствии с леммой 4 алгоритм 2 строит оптимальную последовательность обслуживания требований за время $O(n^7)$.

Процедура может быть легко модифицирована для решения задачи $STRSP2|r_j|C_{max}$. Для этого необходимо изменить строки 4 и 9 процедуры $Sequence$:

4. Вернуть пару $(C_{j_{n_1}} + TAIL_{left}, \sigma)$;

...

9. Вернуть пару $(C_{i_{n_2}} + TAIL_{right}, \sigma)$;

Для решения задачи $1|setup-times, N_1, N_2, p_j = p, r_j| \sum C_j$ в процедуру необходимо внести следующие изменения:

4. Вернуть пару $(\sum_{x:=l+1}^{n_1} C_{j_x}, \sigma)$, где C_{j_x} — время окончания обслуживания требования j_x при частичном расписании, полученном из последовательности σ, где требования обслуживаются с момента времени $S_{i_{k-1}} + p + st_1$;

...

9. Вернуть пару $(\sum_{x:=k}^{n_2} C_{i_x}, \sigma)$, где C_{i_x} — время окончания обслуживания требования i_x при частичном расписании, полученном из последовательности σ, где требования обслуживаются, начиная с $S_{i_{k-1}} + p$;

...

25. **If** $f_{min} > f + f_{current}$ **Then**, *где* $f_{current}$ *— общее время окончания обслуживания требований из последовательности* $(j_{l+1}, \dots, j_{pos}, i_k)$, *обслуживание которых начинается с момента времени* $S_{i_{k-1}} + p$;

26. $f_{min} := f + f_{current}$;

Напомним, что для задачи $1|setup-times, N_1, N_2, p_j = p, r_j|\sum C_j$ требования множеств N_1 и N_2 должны быть упорядочены согласно лемме 4.

Аналогично процедура может быть модифицирована для решения задач $1|setup{-}times, N_1, N_2, p_j = p|\sum w_jC_j$ и $1|setup{-}times, N_1, N_2, p_j = p|\sum T_j$. Заметим, что для этих двух задач $|\Theta| = O(n^3)$, т.к. все моменты поступления работ равны 0, т.е. сложность модифицированных алгоритмов будет равна $O(n^6)$ операций. Таким образом, имеем следующую лемму.

Лемма 5 *С помощью алгоритма 2 и его модификаций следующие задачи могут быть решены за время $O(n^7)$ или $O(n^6)$:*

- $1|setup-times, N_1, N_2, p_j = p, r_j|C_{max}$ *и* $STRSP2|r_j|C_{max}$;

- $1|setup-times, N_1, N_2, p_j = p, r_j \sum C_j$ *и* $STRSP2|r_j|\sum C_j$;

- $1|setup-times, N_1, N_2, p_j = p|\sum w_jC_j$ *и* $STRSP2||\sum w_jC_j$;

- $1|setup-times, N_1, N_2, p_j = p|\sum T_j$ *и* $STRSP2||\sum T_j$.

Задачи с частично упорядоченными подмножествами

Лемма 6 *Для задачи $1|setup-times, N_1, N_2, p_j = p|\sum w_jU_j$ существует оптимальное “сдвинутое влево” расписание, в котором все незапаздывающие требования из одного и того же множества N_1 или N_2 упорядочены по неубыванию директивных сроков, т.е. $d_{j_1} \leq d_{j_2} \leq \cdots \leq d_{j_{n_1}}$ и $d_{i_1} \leq d_{i_2} \leq \cdots \leq d_{i_{n_2}}$.*

Лемма 7 *Предположим, что требования упорядочены в соответствии с леммой 6. Для задачи $1|setup-times, N_1, N_2, p_j = p|\sum U_j$ существует оптимальное “сдвинутое влево” расписание и такие индексы $x,\ 1 \leq x \leq n_1$, и $y, 1 \leq y \leq n_2$, что только требования $j_x, j_{x+1}, \dots, j_{n_1}$, $i_y, i_{y+1}, \dots, i_{n_2}$ не запаздывают и обслуживаются в порядке согласно лемме 6.*

Обе леммы могут быть доказаны по аналогии с леммой 4.

Таким образом, для задачи $1|setup-times, N_1, N_2, p_j = p|\sum U_j$ необходимо найти индексы x и y такие, что $x + y \to \max$ и требования $j_x, j_{x+1}, \dots, j_{n_1}$, $i_y, i_{y+1}, \dots, i_{n_2}$ могут быть выполнены вовремя (без запаздывания) вначале расписания. Следовательно, нужно рассмотреть не более $(n_1+1)\log(n_2+1)$ пар (x,y). Для каждой такой пары решается задача $1|setup-times, N_1, N_2, p_j = p|\sum T_j$ с множеством требований $\{j_x, j_{x+1}, \ldots, j_{n_1}, i_y, i_{y+1}, \ldots, i_{n_2}\}$ с помощью модификации алгоритма 2. Если $\sum T_j = 0$, то пара (x,y) допустима.

Лемма 8 *Задача* $1|setup-times, N_1, N_2, p_j = p|\sum U_j$ *разрешима за время* $O(n^7 \log n)$.

Для задачи $1|setup-times, N_1, N_2, p_j = p|\sum w_j U_j$ предлагается полиномиальный алгоритм динамического программирования. Алгоритм основан на следующих предположениях. Обозначим требования из $N = \{H_1, H_2, \ldots, H_n\}$, где $w_{H_1} \leq w_{H_2} \leq \cdots \leq w_{H_n}$. Если $w_{H_k} = w_{H_{k+1}}$, то $d_{H_k} \leq d_{H_{k+1}}$. Требования из N_1 и N_2 обозначены и упорядочены в соответствии с леммой 6. Пусть $H_n \in N_2$ и $H_n = i_k$. Для H_n позиция в расписании определяется парой (t,l), где $t \in \Theta$ — время начала обслуживания требования, а индекс $l = 0, 1, \ldots, n_1$ означает, что обслуживание незапаздывающих требований из множества $\{j_1, j_2, \ldots, j_l\}$ предшествует обслуживанию требования H_n при расписании, а незапаздывающие требования из множества $\{j_{l+1}, j_{l+2}, \ldots, j_{n_1}\}$ обслуживаются после обслуживания требования H_n. Позиция $(-, n_1+1)$ означает, что требование H_n запаздывает и обслуживается в конце расписания с некоторого момента времени $T \in \Theta$.

Таким образом, для каждой позиции (t,l) среди $O(n^4)$ возможный позиций, мы можем разделить исходную задачу на две независимые подзадачи:

- с множеством требований $N_{left} = \{j_1, j_2, \ldots, j_l, i_1, i_2, \ldots, i_{k-1}\}$, которые должны быть обслужены в интервале $[0, t)$;

- с множеством требований $N_{right} = \{j_{l+1}, j_{l+2}, \ldots, j_{n_1}, i_{k+1}, i_{k+2}, \ldots, i_{n_2}\}$, которые должны быть обслужены в интервале $[t+p, T)$.

Обозначим $T_{max} = \max\{t | t \in \Theta\}$. Отметим, что для задачи $1|setup-times, N_1, N_2, p_j = p|\sum w_j U_j$ мощность множества $|\Theta| = O(n^3)$, поскольку все моменты поступления требований равны 0. Ниже представлен алгоритм решения данной задачи.

Function $SequenceWU(h, t_1, t_2, I_1, I_2, k_1, k_2, l_1, l_2)$

$f_{max} := -\infty$;// взвешенное число незапаздывающих требований;
$\sigma_{max} := \{\}$;
if $H_h \in N_1$ **then**
  $I = 1$;
  if $I_1 = 1$ **then** $t_{min} := t_1$;
  if $I_1 = 2$ **then** $t_{min} := t_1 + st_2$;
  if $I_1 = 0$ **then** $t_{min} := 0$;
  if $I_2 = 1$ **then** $t_{max} := t_2 - p$;
  if $I_2 = 2$ **then** $t_{max} := t_2 - p - st_1$;
  if $I_2 = 0$ **then** $t_{max} := T_{max}$;
  $pos_1 := k_1$; $pos_2 := k_2$;
else
  $I = 2$;
  if $I_1 = 1$ **then** $t_{min} := t_1 + st_1$;
  if $I_1 = 2$ **then**$t_{min} := t_1$;
  if $I_1 = 0$ **then** $t_{min} := 0$;
  if $I_2 = 1$ **then** $t_{max} := t_2 - p - st_2$;
  if $I_2 = 2$ **then** $t_{max} := t_2 - p$;
  if $I_2 = 0$ **then** $t_{max} := T_{max}$;
  $pos_1 := l_1$; $pos_2 := l_2$;
end if
for $pos := pos_1$ to pos_2 **do**
  for each $t \in \Theta,\ t_{min} \le t \le t_{max},\ t + p < d_{H_h}$ **do**
    if $H_h \in N_1$ **then**
      Пусть $j_l = H_h$;
      $(\sigma_1, f_1) := SequenceWU(h-1, t_1, t+p, I_1, I, k_1, pos, l_1, l-1)$;
      $(\sigma_2, f_2) := SequenceWU(h-1, t+p, t_2, I_1, I_2, pos+1, k_2, l+1, l_2)$;
    else
      Пусть $i_k = H_h$;
      $(\sigma_1, f_1) := SequenceWU(h-1, t_1, t+p, I_1, I, k_1, k-1, l_1, pos)$;
      $(\sigma_2, f_2) := SequenceWU(h-1, t+p, t_2, I_1, I_2, k+1, k_2, pos+1, l_2)$;
    end if
    if $f_1 + f_2 + w_{H_h} > f_{max}$ **then**
      $f_{max} := f_1 + f_2 + w_{H_h}$;
      $\sigma_{max} := \sigma_1 \bigcup \sigma_2 \bigcup \{(h, t)\}$;
    end if
  end for
end for
// Дополнительно рассматриваем случай, когда требование H_h запаздывает.
$(\sigma_1, f_1) := SequenceWU(h-1, t_1, t_2, I_1, I_2, k_1, k_2, l_1, l_2)$;
if $f_1 > f_{max}$ **then**
  $f_{max} := f_1$;
  $\sigma_{max} := \sigma_1 \bigcup \{(h, T_{max})\}$;
end if
Вернуть пару (f_{max}, σ_{max});

Несложно оценить трудоемкость данного алгоритма. Множества

Алгоритм 3.
$(F, SCHEDULE_{opt}) := SequenceWU(n, 0, T_{max}, 0, 0, 0, n_2, 0, n_1)$, где $SCHEDULE_{opt}$ — недопустимое расписание, которое может быть трансформировано в оптимальное путем изменения расписания для требований, обслуживание которых назначено с момента времени T_{max}, $F = \sum w_j(1 - U_j)$ — максимальное взвешенное число незапаздывающих требований.

неупорядоченных требований, выступающие в качестве аргументов рекурсивной процедуры, имеют вид

$$N' = \{j_{l_1}, j_{l_1+1}, \ldots, j_{l_2}, i_{k_1}, i_{k_1+1}, \ldots, i_{k_2}\},$$

$$N' \bigcap \{H_{h+1}, H_{h+2}, \ldots, H_n\} = \emptyset,$$

т.е. они однозначно задаются пятью индексами h, k_1, k_2, l_1, l_2. Аргументы t_1, t_2 принадлежат множеству Θ. Таким образом, процедура $SequenceWU(h, t_1, t_2, I_1, I_2, k_1, k_2, l_1, l_2)$ выполняется не более $O(n^{5+3+3})$ раз. Трудоемкость рекурсивной процедуры составляет $O(n^4)$ операций. Следовательно, время работы алгоритма 3 составляет $O(n^{15})$ операций.

Следующая таблица содержит обобщенную информацию по предложенным алгоритмам и задачам, для которых они могут использоваться.

Железнодорожная задача	Соответствующая задача для одного прибора	Сложность алгоритма
$STRSP2\|r_j\|C_{max}$	$1\|setup-times, N_1, N_2, p_j = p, r_j\|C_{max}$	$O(n^7)$
$STRSP2\|r_j\|\sum C_j$	$1\|setup-times, N_1, N_2, p_j = p, r_j \sum C_j$	$O(n^7)$
$STRSP2\|\|\sum w_jC_j$	$1\|setup-times, N_1, N_2, p_j = p\|\sum w_jC_j$	$O(n^6)$
$STRSP2\|\|\sum T_j$	$1\|setup-times, N_1, N_2, p_j = p\|\sum T_j$	$O(n^6)$
$STRSP2\|\|\sum U_j$	$1\|setup-times, N_1, N_2, p_j = p\|\sum U_j$	$O(n^7 \log n)$
$STRSP2\|\|\sum w_jU_j$	$1\|setup-times, N_1, N_2, p_j = p\|\sum w_jU_j$	$O(n^{15})$

Таблица 3: Сложность предложенных алгоритмов для разных типов задач.

Список литературы

[1] P. Brucker, Scheduling Algorithms, Springer-Verlag, 2001.

[2] R.L. Graham , E.L. Lawler, J.K. Lenstra , A.H.G. Rinnooy Kan, Optimization and approximation in deterministic machine scheduling: a survey, Ann. Discrete Math. 5 (1979), 287 –326.

[3] A. Allahverdi , C.T. Ng , T.C.E. Cheng , M.Y. Kovalyov (2008). A survey of scheduling problems with setup times or costs. European Journal of Operational Research, 187, 985–1032.

[4] Ph. Baptiste, P. Brucker, S. Knust and V.G. Timkovsky (2004). Ten notes on equal-processing-time scheduling. 4OR: Quarterly Journal of the Belgian, French and Italian Operations Research Societies, 2, 111 - 127.

[5] S. Kravchenko and F. Werner (2011). Parallel Machine Problems with Equal Processing Times: A Survey. Journal of Scheduling, 14(5), 435 - 444.