УДК 519.8

В. И. САРВАНОВ[1], О. В. ЕФИМОВ[2]

ПОСТРОЕНИЕ РАСПИСАНИЙ ДЛЯ ДВУХСТАДИЙНОЙ СИСТЕМЫ ОБСЛУЖИВАНИЯ ТИПА FLOWSHOP С БЛОКИРОВКАМИ

[1]Институт математики Национальной академии наук Беларуси, Минск, Беларусь, e-mail: sarvanov@im.bas-net.by
[2]ИООО «EPAM Systems», Минск, Беларусь, e-mail: aleh.yafimau@gmail.com

Рассматривается система обслуживания, в которой множество требований $N = N_1 \cup N_2$, $N_1 \cap N_2 = \emptyset$, обслуживается на приборах M_1 и M_2. Особенность системы состоит в том, что время обслуживания требования из $N_1(N_2)$ прибором $M_2(M_1)$ равно нулю и при этом занятый прибор M_1 блокирует доступ к прибору M_2, а занятый прибор M_2 блокирует выход обслуженных требований из системы. Исследуется задача построения расписания, при котором каждое требование из N_1 (N_2) покидает систему не позже заданного директивного срока D_1 (D_2). Доказано, что эта задача является *NP*-трудной и предложен псевдополиномиальный алгоритм ее решения.

Ключевые слова: система обслуживания типа flowshop, блокировка, псевдополиномиальный алгоритм, динамическое программирование, *NP*-трудная проблема, директивные сроки.

V. I. SARVANOV[1], A. V. YAFIMAU[2]

SOLVING A TWO-MASHINE BLOCKING FLOWSHOP SCHEDULING PROBLEM WITH DUE DATES

[1]Institute of Mathematics of the National Academy of Sciences of Belarus, Minsk, Belarus, e-mail: sarvanov@im.bas-net.by
[2]FLLC «EPAM Systems», Minsk, Belarus, e-mail: aleh.yafimau@gmail.com

This work considers solving a particular type of blocking flowshop scheduling problem. In the environment under consideration there are only two machines (denoted as M_1 and M_2). For each job its execution time on one of these machines equals to zero. For each job with a non-zero execution time on machine M_1 (M_2) there is a due date D_1 (D_2 respectively). The authors prove that the problem is *NP*-hard and propose a pseudo-polynomial time algorithm that solves it.

Keywords: flowshop scheduling, blocking, pseudopolynomial algorithm, dynamic programming, *NP*-hard problem, due dates.

Введение. Одна из задач, возникающих при планировании и оперативном управлении технологическими процессами погрузки-разгрузки на железнодорожной станции, в общих чертах может быть сформулирована следующим образом. Имеется система (грузовой фронт), состоящая из железнодорожного пути и складов, расположенных вдоль этого пути. Каждому складу приписано определенное и известное заранее число вагонов, загрузка которых должна быть осуществлена на этом складе, при этом каждый вагон загружается только на одном складе. Путь имеет начало и конец, так что вагоны поступают в систему с начала пути, а покидают ее, проезжая в его конец. Занумеруем склады в том порядке, как они расположены вдоль пути, считая склад, ближайший к началу пути, первым. Тогда вагон может попасть на i-й склад только в том случае, если склады $1, 2, \ldots, i-1$ свободны, т. е. там нет вагонов, а покинуть i-й склад он может только при условии, что свободны склады $i+1$, $i+2, \ldots$. После того как вагон покинет систему, он направляется в назначенный ему формируемый поезд сразу либо после выполнения других технологических операций, длительности которых известны. Известно также время отправления каждого

формируемого состава, тем самым для каждого вагона задан срок, не позже которого он должен покинуть систему, чтобы успеть попасть в формируемый поезд.

Сформулированная практическая задача допускает естественную постановку в терминах теории расписаний (склады – приборы, вагоны – требования, время отправления формируемых поездов – директивные сроки). Настоящая работа посвящена исследованию соответствующей задачи построения допустимых относительно директивных сроков расписаний для случая двух приборов и двух директивных сроков. Показано, что уже при таких ограничениях она является *NP*-трудной и предложен псевдополиномиальный алгоритм ее решения при этих ограничениях.

Определения и постановка задачи. Далее при описании системы обслуживания и расписаний для этой системы мы придерживаемся терминологии из [1, 2].

Множество требований $N = \{1, 2, \ldots, n\}$ необходимо обслужить в системе, состоящей из двух приборов M_1 и M_2. Каждое требование $i \in N$ должно быть обслужено сначала прибором M_1, затем – прибором M_2. Каждый прибор может одновременно обслуживать лишь одно требование. Отсутствует буфер для промежуточного хранения требований при переходе с прибора M_1 на прибор M_2. В результате, если прибор M_1 завершил обслуживание требования j_1, а прибор M_2 занят обслуживанием требования j_2, то j_1 остается на приборе M_1, занимая его до завершения обслуживания требования j_2 и, соответственно, M_1 не может начинать обслуживание еще какого-либо требования. Особенность рассматриваемой задачи заключается в том, что для каждого требования длительность обслуживания на одном из этих приборов пренебрежимо мала, и можно считать ее равной нулю. Тем не менее требование с нулевой длительностью должно посетить соответствующий прибор и, следовательно, если прибор занят, то такое требование должно ждать момента его освобождения. В итоге множество N разбито на два подмножества N_1 и N_2 так, что требования множества N_1 имеют ненулевую длительность обслуживания на приборе M_1, а требования из N_2 – на приборе M_2. Ненулевую длительность обслуживания требования j обозначим через p_j. Прерывания при обслуживании требования прибором запрещены.

Для каждого требования задан директивный срок, к которому необходимо завершить его обслуживание, причем требования, относящиеся к одному и тому же из двух подмножеств N_1 и N_2, имеют один и тот же директивный срок – D_1 и D_2 соответственно.

Необходимо построить расписание обслуживания всех требований, допустимое относительно заданных директивных сроков.

Ввиду отсутствия прерываний для задания расписания достаточно для каждого требования указать моменты начала его обслуживания приборами M_1 и M_2.

Отметим, что отсутствие промежуточного буфера между приборами приводит к тому, что последовательности обслуживания требований приборами M_1 и M_2 совпадают, и расписание обслуживания требований однозначно определяется их перестановкой $\theta = (j_1, \ldots, j_n)$ (см. [2]). Имея перестановку θ, нетрудно для каждого из требований вычислить моменты начала его обслуживания приборами M_1 и M_2. Поэтому далее мы будем отождествлять расписание с соответствующей перестановкой.

Обозначим через $C_j(\theta)$ момент завершения обслуживания требования j при расписании θ. Расписание θ является допустимым относительно заданных директивных сроков, если для любого требования $j \in N_k$ выполняется $C_j(\theta) \le D_k$, $k = 1, 2$.

Сформулированную задачу построения расписания, допустимого относительно двух заданных директивных сроков, будем далее обозначать через (Σ_2, D_1, D_2).

Пусть $C_{\max}(\theta) = \max\{C_j(\theta) \mid j \in N\}$, т. е. $C_{\max}(\theta)$ – момент завершения обслуживания всех требований при расписании θ. Расписание θ^*, доставляющее минимум функции $C_{\max}(\theta)$, будем называть *кратчайшим*. Из дальнейшего будет видно, что кратчайшее расписание может быть построено простым и эффективным алгоритмом.

Если при некотором допустимом расписании θ требование $i \in N_2$ расположено в θ непосредственно после требования $j \in N_1$ и $p_j \ge p_i$, то, поменяв i и j местами, можно перейти к расписанию,

при котором обслуживание требований i и j начинается одновременно. При этом новое расписание останется допустимым, момент завершения обслуживания требования j не изменится, а требования i – уменьшится.

В итоге можно рассматривать расписание не как перестановку отдельных требований, а как последовательность $\Pi = (\pi_1, \pi_2, \ldots, \pi_m)$, каждый элемент π_k которой является либо отдельным требованием $l, l \in N$ либо парой требований (i, j), $i \in N_1, j \in N_2$, где момент начала обслуживания объединенных в пару требований – один и тот же. В первом случае элемент π_k будем называть простым, а требование l – одиночным. Во втором случае элемент π_k назовем составным. Далее используется именно такое представление расписания в виде последовательности $\Pi = (\pi_1, \pi_2, \ldots, \pi_m)$, и эта последовательность называется расписанием.

Свойства и структура допустимых расписаний. Приведем ряд простых свойств допустимых расписаний. Справедливость каждого из них либо сразу вытекает из условий задачи, либо легко доказывается применением стандартного «перестановочного приема». Использование этих свойств позволит ограничиться рассмотрением расписаний специального вида.

Будем для краткости требование i называть ранним, если $d(i) = \min\{D_1, D_2\}$ и, соответственно, поздним, если $d(i) = \max\{D_1, D_2\}$.

Пусть $\Pi = (\pi_1, \pi_2, \ldots, \pi_m)$ – допустимое расписание.

Свойство 1. *Пусть* $\pi_k = i$, $\pi_l = j$, $d(i) \geq d(j)$ *и* $k < l$. *Тогда расписание, полученное из расписания* Π *транспозицией элементов* π_k *и* π_l, *также будет допустимым.*

Отсюда следует, в частности, что можно ограничиться рассмотрением расписаний, согласно которым обслуживание одиночных поздних требований начинается только после завершения обслуживания всех одиночных ранних.

Свойство 2. *Пусть* k, $k \in [2; m]$, *таково, что* $\pi_k = (i, j)$ *либо* $\pi_k = l$, *где* l *– раннее требование. Тогда расписание, полученное из расписания* Π *путем произвольной перестановки элементов* $\pi_1, \pi_2, \ldots, \pi_{k-1}$, *снова будет допустимым.*

Свойство 3. *Пусть* k, $k \in [2; m]$, *таково, что* $\pi_{k-1} = l$, *где* l *– позднее требование. Если* π_k *– составной элемент либо* $\pi_k = i$, *где* i *– позднее требование, то расписание, полученное из расписания* Π *транспозицией элементов* π_{k-1} *и* π_k, *также будет допустимым.*

Свойство 4. *Пусть* $D_1 \leq D_2$, π_k *– простой элемент, а* π_{k+1} *– составной. Тогда транспозиция элементов* π_k, π_{k+1} *оставляет расписание допустимым.*

Для произвольного расписания $\Pi = (\pi_1, \pi_2, \ldots, \pi_m)$ обозначим через $m_1 = m_1(\Pi)$ число его составных элементов, а через $m_2 = m_2(\Pi)$ и $m_3 = m_3(\Pi)$ – число простых элементов, являющихся, соответственно, ранними и поздними требованиями. Введем два типа расписаний специальной структуры. К первому типу (тип 1) отнесем расписания $\Pi = (\pi_1, \pi_2, \ldots, \pi_m)$, структура которых задается следующими условиями:

π_k – составной элемент, если $k = 1, 2, \ldots, m_1$;
π_k – раннее требование, если $k = m_1 + 1, \ldots, m_1 + m_2$;
π_k – позднее требование, если $k = m_1 + m_2 + 1, \ldots, m$.

Расписания второго типа имеют сходную структуру, и различие, по существу, определяется позицией одного составного элемента, называемого далее особым элементом. К этому типу отнесем расписания, которые подчинены следующим условиям:

π_k – составной элемент, если $k = 1, 2, \ldots, m_1 - 1$ или $k = m_1 + m_2$
(в последнем случае π_k– особый элемент);
π_k – раннее требование, если $k = m_1, m_1 + 1, \ldots, m_1 + m_2 - 1$;
π_k – позднее требование, если $k = m_1 + m_2 + 1, \ldots, m$.

Теорема 1. *Если множество допустимых расписаний не пусто, то оно содержит расписание первого либо второго типа.*

Доказательство. Пусть $\Pi = (\pi_1, \pi_2, \ldots, \pi_m)$ – произвольное допустимое расписание. Обозначим через t такой наибольший индекс, что π_t – составной элемент либо π_t – раннее требование (т. е. простой элемент, являющийся ранним требованием). Рассмотрим эти две возможности для π_t по отдельности, учитывая, что согласно выбору индекса t каждый элемент π_l, для $l > t$, является поздним требованием. Если π_t – раннее требование, то, используя сначала свойство 2, получим допустимое расписание, в котором составные элементы занимают первые m_1 позиций. После этого, выполнив на основе свойства 1 подходящие транспозиции, получим расписание типа 1.

Если π_t – составной элемент, то с использованием свойства 2 сначала получим расписание, в котором на первых $m_1 - 1$ позициях размещены составные элементы (все, кроме π_t). Затем, пользуясь свойствами 1 и 3, придем к расписанию, в котором все простые элементы, являющиеся поздними требованиями, займут позиции правее t. В результате получаем расписание типа 2, если множество элементов расписания П, являющихся ранними требованиями, не пусто, и расписание типа 1 – в противном случае. Теорема 1 доказана.

Расписание П типа 1 будем схематически представлять в виде $\Pi = (\Pi_1, \Pi_2, \Pi_3)$, где Π_1 – отрезок последовательности П, все элементы которого являются составными, а элементы отрезков Π_2 и Π_3, соответственно, – ранними и поздними одиночными требованиями.

Похожее представление будем использовать и для расписаний второго типа. Схематически такое расписание П будем записывать в виде $\Pi = (\Pi_1, \Pi_2, \pi', \Pi_3)$, где Π_2 и Π_3 имеют тот же смысл, что и выше, а отрезок Π_1 содержит все составные элементы расписания П, кроме особого элемента $\pi' = \pi_{m_1+m_2}$. Заметим, что из определения расписания типа 2 следует, что $\Pi_2 \neq \varnothing$.

Если расписание П (любого из двух типов) не содержит элементов какого-либо из трех упомянутых видов, то будем считать соответствующий отрезок Π_i, $i = 1, 2, 3$, пустым и использовать в этом случае обозначение $\Pi_i = \varnothing$.

Из свойств 1, 2 и определения отрезков Π_1, Π_2 и Π_3 в расписаниях П обоих типов вытекает справедливость следующих двух утверждений.

Утверждение 1. *Если в допустимом расписании* П *выполнить произвольную перестановку элементов отрезка* Π_2 (Π_3)*, то снова получим допустимое расписание.*

Утверждение 2. *Если* $\Pi_2 \neq \varnothing$*, то любая перестановка элементов отрезка* Π_1 *оставляет расписание* П *допустимым, а если* $\Pi_2 = \varnothing$*, то расписание остается допустимым при любой перестановке элементов* $\pi_1, \pi_2, \ldots, \pi_{m_1-1}$*, т. е. при перестановке, не затрагивающей последний элемент отрезка* Π_1.

Из теоремы 1 и свойства 4 легко получить

Утверждение 3. *Если* $D_1 \leq D_2$*, то множество допустимых расписаний, когда оно не пусто, содержит расписание типа* 1.

Пусть обслуживание начинается в момент времени 0 и пусть t_1 – момент времени, когда последнее из требований, скажем l, входящих в элементы отрезка Π_1, покидает систему. По условию задачи можно считать, что $l \in N_1$. Кроме того, t_1 является суммой величин $t(\pi_k)$, $k = 1, 2, \ldots, m_1$, где $t(\pi_k)$ – длина интервала времени от начала обслуживания элемента π_k до момента, когда начинается обслуживание элемента π_{k+1}. Таким образом, при любой перестановке элементов отрезка Π_1 величина t_l не меняется. Если расписание $\Pi = (\Pi_1, \Pi_2, \Pi_3)$ является допустимым и $D_1 \leq D_2$, то, с учетом утверждения 2, приходим к следующему.

Утверждение 4. *Пусть* $D_1 \leq D_2$ *и* $\Pi = (\Pi_1, \Pi_2, \Pi_3)$ *– допустимое расписание. Тогда любая перестановка элементов отрезка* Π_1 *приводит к допустимому расписанию.*

Следующее утверждение, по-видимому, является «фольклорным». Его несложное доказательство, использующее стандартный «перестановочный прием», можно найти, например, в работе [2]. Оно показывает, как устроены элементы отрезка Π_1 в допустимых расписаниях обоих типов.

Утверждение 5. *Пусть пара векторов* $x=(x_1,x_2,\ldots,x_p)$ *и* $y=(y_1,y_2,\ldots,y_p)$ *и подстановка* $\tau,\tau\in S_p$, *таковы, что выполняются неравенства:*

$$x_1\le x_2\le\ldots\le x_p \text{ и } y_{\tau(1)}\le y_{\tau(2)}\le\ldots\le y_{\tau(p)}.$$

Тогда для любой подстановки θ, $\theta\in S_p$, *справедливо неравенство*

$$\sum_{i=1}^{p}\max\left(x_i,y_{\theta(i)}\right)\ge\sum_{i=1}^{p}\max\left(x_i,y_{\tau(i)}\right).$$

Согласно последнему утверждению, минимизировать общее время обслуживания (т. е. построить кратчайшее расписание) можно при помощи упорядочивания элементов по длительностям обслуживания.

Обозначим $d=\min\{D_1,D_2\}$, $D=\max\{D_1,D_2\}$, $D_1\ne D_2$. Если общее время обслуживания в кратчайшем расписании располагается вне интервала $(d,D]$, то сразу можно ответить на вопрос о существовании допустимого расписания.

Легко видеть, что кратчайшее расписание можно рассматривать как расписание, допустимое относительно директивных сроков $d(i)=D$, $i\in N$, при соответствующем выборе значения D. Поэтому задача построения кратчайшего расписания полиномиально сводится к задаче построения допустимого, а обратное, вообще говоря, неверно.

Сложностной статус задачи.

Теорема 2. *Задача* (Σ_2,D_1,D_2) *построения допустимого расписания в системе с двумя приборами является NP-трудной.*

Доказательство.

Рассмотрим одну из основных *NP*-полных задач – задачу РАЗБИЕНИЕ.

УСЛОВИЕ. Заданы конечное множество A и «вес» $s(a)\in Z^+$ для каждого $a\in A$.

ВОПРОС. Существует ли подмножество $A'\subseteq A$, такое, что

$$\sum_{a\in A'}s(a)=\sum_{a\in A\setminus A'}s(a)?$$

Построим полиномиальное сведение задачи РАЗБИЕНИЕ к нашей задаче.

Пусть в задаче РАЗБИЕНИЕ $|A|=n$, $n>0$. Положим $A=\{1,\ldots,n\}$ и $s(i)=a_i$, $a_i\in Z^+$, $i=\overline{1,n}$. Выберем параметр δ из условия

$$\delta=n\cdot\max_{1\le i\le n}\{a_i\}.$$

Определим множества требований и длительности их обслуживания следующим образом:

$$N_1=\left\{i\,\middle|\,p_i=i\delta,i=\overline{1,n}\right\},\ N_2=\left\{(2j-1),2j\,\middle|\,p_{2j-1}=j\delta-a_j;p_{2j}=j\delta+a_j;j=\overline{1,n}\right\}.$$

Положим

$$D_1=\delta+2\delta+\ldots+n\delta+\frac{1}{2}\sum_{i=1}^{n}a_i,\ \text{а}\ D_2=D_1+\delta+2\delta+\ldots+n\delta\ (\text{т. е. } D_1<D_2)$$

и примем D_i, $i=1,2$, в качестве директивного срока для требований из множества N_i, $i=1,2$. Очевидно, что такое сведение является полиномиальным.

Покажем теперь, что задача РАЗБИЕНИЕ имеет решение тогда и только тогда, когда существует допустимое расписание для построенного примера рассматриваемой задачи.

Необходимость. Предположим, что существует подмножество $A'\subseteq A$ такое, что

$$\sum_{a\in A'}s(a)=\sum_{a\in A\setminus A'}s(a)=\frac{1}{2}\sum_{i=1}^{n}a_i.$$

Тогда можно построить расписание первого типа, имеющее вид:

$$\Pi_1 = \{\pi_k\}, \text{ где } \begin{cases} \pi_k = (k, 2k), \text{где } p_k = k\delta,\ p_{2k} = k\delta + a_k \text{ при } k \in A' \\ \pi_k = (k, 2k-1), \text{где } p_k = k\delta,\ p_{2k-1} = k\delta - a_k \text{ при } k \in A \setminus A', \end{cases}$$

$$\Pi_2 = \varnothing,$$

$$\Pi_3 = \{\pi_k\}, \text{ где } \begin{cases} \pi_k = (2k), \text{где } p_{2k} = k\delta + a_k \text{ при } k \in A \setminus A' \\ \pi_k = (2k-1), \text{где } p_{2k-1} = k\delta - a_k \text{ при } k \in A'. \end{cases}$$

Обслуживание элемента π_k из отрезка Π_1 потребует времени $k\delta + a_k$, если $\pi_k = (k, 2k)$, и времени $k\delta$, если $\pi_k = (k, 2k-1)$. Тогда обслуживание последнего элемента на отрезке Π_1 завершится к моменту времени

$$\sum_{k \in A'} (k\delta + a_k) + \sum_{k \in A \setminus A'} k\delta = \sum_{k \in A} k\delta + \sum_{k \in A'} a_k = \delta + 2\delta + \ldots + n\delta + \frac{1}{2}\sum_{i=1}^{n} a_i = D_1.$$

Обслуживание элемента π_k из отрезка Π_3 потребует времени $k\delta + a_k$, если $\pi_k = (2k)$, и времени $k\delta - a_k$, если $\pi_k = (2k-1)$. Тогда обслуживание последнего элемента на отрезке Π_3 завершится к моменту времени

$$D_1 + \sum_{k \in A'} (k\delta + a_k) + \sum_{k \in A \setminus A'} (k\delta - a_k) = D_1 + \sum_{k \in A} k\delta + \sum_{k \in A'} a_k - \sum_{k \in A \setminus A'} a_k = D_1 + \sum_{k \in A} k\delta = D_2.$$

Таким образом, построенное расписание является допустимым.

Достаточность. Предположим, что для построенного примера нашей задачи существует допустимое расписание. Установим свойства такого расписания.

Лемма 1. *Допустимое расписание является расписанием первого типа, в котором*

$\Pi_1 = \{\pi_k = (k, j)\}$, *причем,* $j = 2k$, *где* $p_{2k} = k\delta + a_k$, *либо* $j = 2k-1$, *где* $p_{2k-1} = k\delta - a_k$,

$\Pi_2 = \varnothing$,

$\Pi_3 = \{\pi_k\}$, *причем,* $\pi_k = (2k)$, *где* $p_{2k} = k\delta + a_k$, *либо* $\pi_k = (2k-1)$, *где* $p_{2k-1} = k\delta - a_k$.

Докажем это утверждение.

Предположим, что можно построить расписание типа 2, тогда в силу данных определений $\Pi_2 \neq \varnothing$, т. е. $\exists i_0 \in N_1$, что $(i_0) \in \Pi_2$. Так как каждое требование из множества N_2 должно быть обслужено, то общее время обслуживания этих требований не менее чем $(\delta - a_1) + (\delta + a_1) + \ldots + (n\delta - a_n) + (n\delta + a_n)$. Кроме того, требование i_0 не входит в пару с каким-либо требованием. Поэтому его обслуживание потребует времени $i_0\delta$. Тогда общее время обслуживания всех требований не менее чем

$$(\delta - a_1) + (\delta + a_1) + \ldots + (n\delta - a_n) + (n\delta + a_n) + i_0\delta \geq [i_0 \geq 1] \geq$$

$$\geq 2(\delta + \ldots + n\delta) + \delta > \left[\delta > \frac{1}{2}\sum_{i=1}^{n} a_i\right] > 2(\delta + \ldots + n\delta) + \frac{1}{2}\sum_{i=1}^{n} a_i = D_2.$$

Получаем противоречие с допустимостью расписания.

Предположим, что в построенном допустимом расписании $\exists k_0 \in N_1$, что $\pi_{k_0} = (k_0, j_0) \in \Pi_1$, причем $j_0 \neq 2k_0$ и $j_0 \neq 2k_0 - 1$. Пусть $j_0 = 2l_0$ либо $j_0 = 2l_0 - 1\,(l_0 \neq k_0)$. Рассмотрим два случая.

а) $l_0 < k_0$.

Пусть для определенности $j_0 = 2l_0$ (случай $j_0 = 2l_0 - 1$ доказывается аналогично). Тогда для обслуживания всех требований из множества N_2, за исключением $(2l_0)$, потребуется, по крайней мере, время

$$(\delta - a_1) + (\delta + a_1) + \ldots + [(l_0 - 1)\delta - a_{l_0-1}] + [(l_0 - 1)\delta + a_{l_0-1}] + (l_0\delta - a_{l_0}) + [(l_0 + 1)\delta - a_{l_0+1}] +$$
$$+ [(l_0 + 1)\delta + a_{l_0+1}] + \ldots + (n\delta - a_n) + (n\delta + a_n).$$

Так как требование $(2l_0)$ обслуживается в паре с требованием k_0 и $p_{2l_0} = l_0\delta + a_{l_0}$, $p_{k_0} = k_0\delta$, то для обслуживания этой пары потребуется время $k_0\delta$, ибо

$$l_0\delta + a_{l_0} < l_0\delta + \delta \le [l_0 + 1 \le k_0] \le k_0\delta.$$

Тогда общее время обслуживания всех упомянутых требований составляет

$$(\delta - a_1) + (\delta + a_1) + \ldots + [(l_0 - 1)\delta - a_{l_0-1}] + [(l_0 - 1)\delta + a_{l_0-1}] + (l_0\delta - a_{l_0}) + k_0\delta +$$
$$+ [(l_0 + 1)\delta - a_{l_0+1}] + [(l_0 + 1)\delta + a_{l_0+1}] + \ldots + (n\delta - a_n) + (n\delta + a_n) = [k_0\delta = (l_0\delta + a_{l_0}) + ((k_0 - l_0)\delta - a_{l_0})] =$$
$$= 2(\delta + \ldots + n\delta) + ((k_0 - l_0)\delta - a_{l_0}) \ge [k_0 - l_0 \ge 1] \ge 2(\delta + \ldots + n\delta) + (\delta - a_{l_0}) > \left[\delta - a_{l_0} > \frac{1}{2}\sum_{i=1}^{n} a_i\right] >$$
$$> 2(\delta + \ldots + n\delta) + \frac{1}{2}\sum_{i=1}^{n} a_i = D_2.$$

Получаем противоречие с допустимостью расписания.

б) $l_0 > k_0$.

Здесь, для определенности, положим $j_0 = 2l_0 - 1$ (случай $j_0 = 2l_0$ доказывается аналогично). Для обслуживания всех требований из множества N_1, за исключением (k_0), потребуется, по крайней мере, время $\delta + \ldots + (k_0 - 1)\delta + (k_0 + 1)\delta + \ldots + n\delta$. Так как требования (k_0) и $(2l_0 - 1)$ обслуживаются в паре и $p_{k_0} = k_0\delta$, $p_{2l_0-1} = l_0\delta - a_{l_0}$, то для обслуживания этой пары потребуется время $l_0\delta - a_{l_0}$, ибо $l_0\delta - a_{l_0} > [a_{l_0} < \delta] > l_0\delta - \delta \ge [l_0 - 1 \ge k_0] \ge k_0\delta$. Учитывая время обслуживания пары $(k_0, 2l_0 - 1)$, обслуживание компонент из отрезка Π_1 завершится не ранее, чем к моменту времени

$$\delta + \ldots + (k_0 - 1)\delta + l_0\delta - a_{l_0} + (k_0 + 1)\delta + \ldots + n\delta = [l_0\delta = k_0\delta + (l_0 - k_0)\delta] = (\delta + \ldots + n\delta) +$$
$$+ (l_0 - k_0)\delta - a_{l_0} \ge [l_0 - k_0 \ge 1] \ge (\delta + \ldots + n\delta) + \delta - a_{l_0} > \left[\delta - a_{l_0} > \frac{1}{2}\sum_{i=1}^{n} a_i\right] > (\delta + \ldots + n\delta) + \frac{1}{2}\sum_{i=1}^{n} a_i = D_1.$$

Противоречие с допустимостью расписания.

Таким образом, лемма 1 доказана.

Лемма 2. *Расписание является допустимым тогда и только тогда, когда обслуживание компонент отрезка* Π_1 *завершается в момент времени* D_1.

Введем следующие обозначения. Учитывая структуру расписания согласно свойству 1, обозначим через I множество индексов $\{1,\ldots,n\}$, а через I' – его подмножество, а именно: $I' = \{i \mid i \in I, (i, 2i) \in \Pi_1,$ где $p_{2i} = i\delta + a_i\}$. В дальнейших рассуждениях будем также опираться на структуру расписания согласно свойству 1.

Достаточность леммы 2 почти очевидна: все требования с директивным сроком D_1 входят в компоненты отрезка Π_1. Поскольку

$$\sum_{i \in I \setminus I'} i\delta + \sum_{i \in I'} (i\delta + a_i) = D_1, \text{ то } \sum_{i \in I} i\delta + \sum_{i \in I'} a_i = D_1,$$

$$\text{т. е. } \sum_{i \in I'} a_i = \frac{1}{2}\sum_{i=1}^{n} a_i \text{ и } \sum_{i \in I \setminus I'} a_i = \sum_{i \in I} a_i - \frac{1}{2}\sum_{i=1}^{n} a_i = \frac{1}{2}\sum_{i=1}^{n} a_i.$$

Тогда обслуживание всех требований закончится к моменту

$$D_1 + \sum_{i \in I'} (i\delta - a_i) + \sum_{i \in I \setminus I'} (i\delta + a_i) = D_1 + \sum_{i \in I} i\delta - \sum_{i \in I'} a_i + \sum_{i \in I \setminus I'} a_i = D_1 + \sum_{i \in I} i\delta = D_2.$$

Таким образом, расписание допустимо.

Необходимость докажем методом от противного. Предположим, что существует допустимое расписание, в котором обслуживание компонент отрезка Π_1 завершается ранее момента времени D_1, т. е.

$$\sum_{i\in I\setminus I'} i\delta + \sum_{i\in I'} \left(i\delta + a_i\right) < D_1,$$

что равносильно неравенству

$$\sum_{i\in I} i\delta + \sum_{i\in I'} a_i < D_1 = \sum_{i\in I} i\delta + \frac{1}{2}\sum_{i=1}^{n} a_i,$$

из которого следует, что

$$\sum_{i\in I'} a_i < \frac{1}{2}\sum_{i=1}^{n} a_i \text{ и } \sum_{i\in I\setminus I'} a_i = \sum_{i\in I} a_i - \sum_{i\in I'} a_i > \sum_{i\in I} a_i - \frac{1}{2}\sum_{i=1}^{n} a_i = \frac{1}{2}\sum_{i=1}^{n} a_i.$$

Тогда общее время обслуживания составляет

$$\sum_{i\in I} i\delta + \sum_{i\in I'} a_i + \sum_{i\in I'} \left(i\delta - a_i\right) + \sum_{i\in I\setminus I'} \left(i\delta + a_i\right) =$$

$$= \sum_{i\in I} i\delta + \sum_{i\in I'} a_i + \sum_{i\in I} i\delta - \sum_{i\in I'} a_i + \sum_{i\in I\setminus I'} a_i = 2\sum_{i\in I} i\delta + \sum_{i\in I\setminus I'} a_i > 2\sum_{i\in I} i\delta + \frac{1}{2}\sum_{i=1}^{n} a_i = D_2.$$

Последнее противоречит допустимости расписания. Лемма 2 доказана.

Итак, если существует допустимое расписание, то $\sum_{i\in I'} a_i = \frac{1}{2}\sum_{i=1}^{n} a_i$, тогда в качестве $A' \subseteq A$ для задачи РАЗБИЕНИЕ можно взять множество I'. Таким образом, *NP*-трудность задачи построения допустимого расписания доказана полностью.

Можно доказать, что задача $\left(\Sigma_2, D_1, D_2\right)$ остается *NP*-трудной и в случае, когда $D_2 < D_1$. Схема соответствующего доказательства выглядит следующим образом. Построим сведение задачи РАЗБИЕНИЕ аналогично случаю $D_1 < D_2$, с тем лишь отличием, что роли приборов инвертированы. Определим множества требований и длительности их обслуживания следующим образом:

$$\tilde{N}_1 = \left\{ \left(2j-1\right), 2j \middle| p_{2j-1} = j\delta - a_j;\ p_{2j} = j\delta + a_j;\ j = \overline{1,n} \right\},\ \tilde{N}_2 = \left\{ i \middle| p_i = i\delta, i = \overline{1,n} \right\}.$$

Кроме того, дополним множество требований на прибор M_1 (M_2) одним требованием с длительностью обслуживания $\left(n+2\right)\delta$ $\left(\left(n+1\right)\delta\right)$, полагая:

$$N_1 = \tilde{N}_1 \cup \left\{ \left(2n+1\right) \middle| p_{2n+1} = \left(n+2\right)\delta \right\},\ N_2 = \tilde{N}_2 \cup \left\{ \left(n+1\right) \middle| p_{n+1} = \left(n+1\right)\delta \right\}.$$

Определим директивные сроки D_1 и D_2 следующим образом:

$$D_2 = \delta + 2\delta + \ldots + n\delta + \frac{1}{2}\sum_{i=1}^{n} a_i + \left(n+1\right)\delta,\ D_1 = D_2 + \left(\delta + 2\delta + \ldots + n\delta\right) + \delta \text{ (т. е. } D_2 < D_1\text{)}.$$

Можно показать, что если допустимое расписание существует, то требования $\left(2n+1\right)$ $\left(\text{где } p_{2n+1} = \left(n+2\right)\delta\right)$ и $\left(n+1\right)$ $\left(\text{где } p_{n+1} = \left(n+1\right)\delta\right)$ могут обслуживаться только в паре, и эта пара является особым элементом.

Теперь, когда особый элемент «зафиксирован», доказательство *NP*-трудности задачи для случая $D_2 < D_1$ проводится точно так же, как и в случае $D_1 < D_2$.

Псевдополиномиальный алгоритм. Введем обозначения для мощностей множеств N_1 и N_2: $n_1 = \left|N_1\right|, n_2 = \left|N_2\right|$. Таким образом, $n_1 + n_2 = n$. Обозначим через $a_i \left(i = \overline{1,n_1}\right)$ и $b_j \left(j = \overline{1,n_2}\right)$ длительности обслуживания требований множеств N_1 и N_2 соответственно. Будем считать (не ограничивая общности), что $a_{i_1} \geq a_{i_2} \left(i_1 < i_2\right), b_{j_1} \geq b_{j_2} \left(j_1 < j_2\right)$.

Вначале рассмотрим случай, когда $D_1 < D_2$ и, согласно утверждению 3, существует расписание типа 1.

Рассмотрим произвольное допустимое расписание П. Пусть для некоторого элемента $(i_1, j_1) \in \Pi_1$ и элемента $i_2 \in \Pi_2$ выполняется соотношение $i_2 < i_1$ (т. е. $a_{i_2} > a_{i_1}$). Тогда замена элемента (i_1, j_1) на (i_2, j_1) и одновременно элемента i_2 на i_1 не нарушит допустимости расписания, ибо $\max(a_{i_1}, b_{j_1}) + a_{i_2} \geq \max(a_{i_2}, b_{j_1}) + a_{i_1}$. В связи с этим будем рассматривать лишь такие расписания, в которых $\forall(i_1, j_1) \in \Pi_1$ и $\forall i_2 \in \Pi_2$ выполняется $a_{i_1} > a_{i_2}$. Их далее будем называть каноническими расписаниями. Кроме того, согласно утверждению 4 можно переупорядочить требования на отрезке Π_1, сохраняя допустимость расписания. Из приведенных ранее утверждений легко получить следующие свойства допустимого расписания канонического вида:

1) ранние требования обслуживаются в порядке невозрастания длительностей обслуживания;

2) на каждом из отрезков Π_1 и Π_3 поздние требования обслуживаются в порядке невозрастания длительностей обслуживания.

Таким образом, задача сводится к отысканию «назначения», т. е. формирования пар из ранних и поздних требований, составляющих отрезок Π_1, так чтобы все требования множества N были обслужены согласно директивным срокам.

Так как требования начинают обслуживаться в нулевой момент времени, то обслуживание последней пары требований на отрезке Π_1 завершится к моменту $\sum_{(i,j)\in\pi_1} \max(a_i, b_j)$, последнего требования на отрезке Π_2 – к моменту

$$\sum_{(i,j)\in\Pi_1} \max(a_i, b_j) + \sum_{i\in\Pi_2} a_i,$$

а последнего требования на отрезке Π_3 – к моменту

$$\sum_{(i,j)\in\Pi_1} \max(a_i, b_j) + \sum_{i\in\Pi_2} a_i + \sum_{j\in\Pi_3} b_j.$$

Согласно условиям допустимости расписания, должны выполняться следующие неравенства (неравенства допустимости):

$$\sum_{(i,j)\in\Pi_1} \max(a_i, b_j) + \sum_{i\in\Pi_2} a_i \leq D_1, \quad \sum_{(i,j)\in\Pi_1} \max(a_i, b_j) + \sum_{i\in\Pi_2} a_i + \sum_{j\in\Pi_3} b_j \leq D_2.$$

Положим, что $A = \sum_{i=1}^{n_1} a_i$, т. е. A – величина, непосредственно определяемая на основании входных параметров задачи. Тогда неравенства допустимости можно переписать в виде:

$$\sum_{\substack{(i,j)\in\Pi_1 \\ b_j > a_i}} b_j \leq D_1 - A, \quad \sum_{\substack{(i,j)\in\Pi_1 \\ b_j > a_i}} b_j + \sum_{j\in\Pi_3} b_j \leq D_2 - A.$$

Сразу отметим, что в случае $D_1 < A$ задача решения не имеет. Таким образом, можно считать, что правые части указанных неравенств неотрицательны.

Положим далее, что $B = \sum_{j=1}^{n_2} b_j$. Аналогично величине A, величина B непосредственно определяется на основании входных параметров задачи. Отметим также, что множество поздних требований, которые представлены в элементах отрезков Π_1 и Π_3, это в точности множество N_2. Потому выполняется равенство $\sum_{(i,j)\in\Pi_1} b_j + \sum_{j\in\Pi_3} b_j = B$.

Тогда второе неравенство допустимости можно записать в виде

$$\sum_{\substack{(i,j)\in\Pi_1 \\ b_j > a_i}} b_j + B - \sum_{(i,j)\in\Pi_1} b_j \leq D_2 - A, \text{ что равносильно } \sum_{\substack{(i,j)\in\Pi_1 \\ b_j \leq a_i}} b_j \geq A + B - D_2.$$

Заметим, что в случае, когда $A+B \le D_2$, задача является тривиальной (если $A \le D_1$, то существование допустимого расписания очевидно, а в противном случае оно не существует). Поэтому нас интересует лишь случай, когда $A+B-D_2 > 0$.

Итак, требуется найти такое «назначение» в пары (i,j), составляющие отрезок Π_1, чтобы выполнялись оба неравенства:

$$\sum_{\substack{(i,j)\in\Pi_1 \\ b_j > a_i}} b_j \le D_1 - A \quad \text{и} \quad \sum_{\substack{(i,j)\in\Pi_1 \\ b_j \le a_i}} b_j \ge A + B - D_2.$$

Будем решать задачу максимизации суммы во втором неравенстве при ограничении на величину суммы в первом неравенстве. Для этого воспользуемся методом динамического программирования. В качестве подзадачи рассмотрим задачу, в которой заданы k поздних требований (т. е. $b_1,\cdots,b_k$), требуется образовать l пар, и $\sum_{\substack{(i,j)\in\Pi_1 \\ b_j > a_i}} b_j = S$.

Рассмотрим функционал $F(k,l,S)$, определенный для $k = 0,...,n_2$, $l = 0,...,\min(n_1,n_2)$, $S = 0,...,(D_1 - A)$. $F(k,l,S)$ равно наибольшему значению $\sum_{\substack{(i,j)\in\Pi_1 \\ b_j \le a_i}} b_j$ при условии, что рассматривается всего k значений b_j, образовано l пар (с $a_1,\cdots,a_l$) и $\sum_{\substack{(i,j)\in\Pi_1 \\ b_j > a_i}} b_j = S$. Тогда справедливо следующее рекуррентное соотношение:

$$F(k,l,S) = \max\left\{ F(k-1,l,S),\quad \underset{\text{только когда } b_k > a_l}{F(k-1,l-1,S-b_k)},\quad \underset{\text{только когда } b_k \le a_l}{F(k-1,l-1,S)-b_k} \right\}.$$

Заполнение таблицы будем производить по следующим формулам:

$F(k+1,l,S) = \max\{F(k+1,l,S), F(k,l,S)\}$ (когда новая пара не образовывается);
$F(k+1,l+1,S+b_{k+1}) = \max\{F(k+1,l+1,S+b_{k+1}), F(k,l,S)\}$ (если $b_{k+1} > a_{l+1}$);
$F(k+1,l+1,S) = \max\{F(k+1,l+1,S), F(k,l,S)+b_{k+1}\}$ (если $b_{k+1} \le a_{l+1}$).

Начальные условия: $F(0,0,0) = 0$. Во всех остальных ячейках таблицы записаны значения –1. Для перехода к конечному состоянию (после того, как таблица заполнена) ищем максимум значений F по нижнему уровню таблицы (n_2,l,S), где $l = 0,...,\min(n_1,n_2)$, $S = 0,...,(D_1 - A)$.

Если на нижнем уровне нашлись такие значения l_0 и S_0, что $F(n_2,l_0,S_0) \ge A+B-D_2$, то существует допустимое расписание.

Заполнение таблицы происходит по столбцам в порядке возрастания значения уровня k, на каждом уровне столбцы обходятся в направлении увеличения значений l:

$$k = 0,...,n_2,$$
$$l = 0,...,\min(n_1,n_2),$$
$$S = 0,...,(D_1 - A).$$

Для заполнения значениями $F(k_0,l,S)$ на некотором уровне k_0 достаточно хранить в памяти две двумерные таблицы: для значений $F(k_0-1,l,S)$ и $F(k_0,l,S)$. Трудоемкость алгоритма определяется затратами времени на построение таблицы и составляет $O(n_2 \cdot \min(n_1,n_2) \cdot (D_1 - A))$.

Рассмотрим теперь случай, когда $D_2 < D_1$, и покажем, как модифицировать изложенный выше алгоритм, чтобы получить решение задачи в этой ситуации. Так как и в случае $D_2 < D_1$ может оказаться, что существует допустимое расписание типа 1, то попробуем сначала построить расписание этого типа по аналогии со случаем $D_1 < D_2$.

Обслуживание последней пары требований на отрезке Π_1 завершится к моменту $\sum_{(i,j)\in\Pi_1} \max\left(a_i, b_j\right)$, последнего требования на отрезке Π_2 – к моменту $\sum_{(i,j)\in\Pi_1} \max\left(a_i, b_j\right) + \sum_{j\in\Pi_2} b_j$, а последнего требования на отрезке Π_3 – к моменту $\sum_{(i,j)\in\Pi_1} \max\left(a_i, b_j\right) + \sum_{j\in\Pi_2} b_j + \sum_{i\in\Pi_3} a_i$.

Согласно условиям допустимости расписания, должны выполняться неравенства

$$\sum_{(i,j)\in\Pi_1} \max\left(a_i, b_j\right) + \sum_{j\in\Pi_2} b_j \le D_2, \quad \sum_{(i,j)\in\Pi_1} \max\left(a_i, b_j\right) + \sum_{j\in\Pi_2} b_j + \sum_{i\in\Pi_3} a_i \le D_1.$$

Следуя аналогичным определениям величин A и B, неравенства допустимости можно переписать в виде:

$$\sum_{\substack{(i,j)\in\Pi_1 \\ a_i > b_j}} \left(a_i - b_j\right) \le D_2 - B, \quad \sum_{\substack{(i,j)\in\Pi_1 \\ a_i > b_j}} \left(a_i - b_j\right) + \sum_{i\in\Pi_3} a_i \le D_1 - B.$$

Далее по аналогии со случаем $D_1 < D_2$ второе неравенство допустимости при помощи равносильных преобразований можно привести к виду $\sum_{(i,j)\in\Pi_1} \min\left(a_i, b_j\right) \ge A + B - D_1$.

Итак, требуется найти такое «назначение» в пары (i, j) на отрезке Π_1, чтобы выполнялись оба неравенства:

$$\sum_{\substack{(i,j)\in\Pi_1 \\ a_i > b_j}} \left(a_i - b_j\right) \le D_2 - B, \quad \sum_{(i,j)\in\Pi_1} \min\left(a_i, b_j\right) \ge A + B - D_1.$$

Будем решать задачу максимизации суммы во втором неравенстве при ограниченной сумме в первом неравенстве. Тогда, применяя метод динамического программирования, в качестве подзадачи рассмотрим задачу, в которой заданы k поздних требований (т. е. $a_1, \cdots, a_k$), требуется образовать l пар, и

$$\sum_{\substack{(i,j)\in\Pi_1 \\ a_i > b_j}} \left(a_i - b_j\right) = S.$$

Рассмотрим функционал $G(k,l,S)$, определенный для $k = 0,...,n_1$, $l = 0,...,\min\left(n_1, n_2\right)$, $S = 0,...,\left(D_2 - B\right)$. $G(k,l,S)$ равно наибольшему значению $\sum_{(i,j)\in\Pi_1} \min\left(a_i, b_j\right)$ при условии, что рассматривается всего k значений a_i, образовано l пар (с $b_1, \cdots, b_l$) и $\sum_{\substack{(i,j)\in\Pi_1 \\ a_i > b_j}} \left(a_i - b_j\right) = S$. Тогда справедливо следующее рекуррентное соотношение:

$$G(k,l,S) = \max\left\{ G\left(k-1,l,S\right), \quad \underset{\text{только когда } a_k > b_l}{G\left(k-1,l-1,S-\left(a_k - b_l\right)\right) + b_l}, \quad \underset{\text{только когда } a_k \le b_l}{G\left(k-1,l-1,S\right) + a_k} \right\}.$$

Формулы для заполнения таблицы, в которой хранятся значения функционала $G(k,l,S)$, строятся по аналогии с формулами для таблицы, в которой хранятся значения функционала $F(k,l,S)$.

Если допустимое расписание типа 1 построить не удалось, то предположим существование допустимого расписания типа 2. Для особого элемента, представленного парой $\left(a_{i_0}, b_{j_0}\right)$ в допустимом расписании типа 2, должно выполняться неравенство $a_{i_0} > b_{j_0}$, иначе рассматриваемый элемент не был бы особым.

На основании входных данных исходной задачи построим модифицированную задачу. В этой задаче множество ранних требований не содержит элемента с длительностью обслуживания b_{j_0}, множество поздних требований не содержит элемента с длительностью обслуживания a_{i_0},

а директивные сроки изменены следующим образом: $D_2' = D_2 \quad b_{j\bar{0}}$, $D_1' = D_1 \quad a_{i\bar{0}}$. Тогда если существует допустимое расписание типа 1 для модифицированной задачи, то можно очевидным образом перейти к расписанию типа 2 для исходной задачи. Таким образом, нужно последовательно фиксировать индекс j_0 раннего требования и в качестве пары для особого элемента рассматривать такие индексы i_0 поздних требований, что $a_{i0} > b_{j0}$. Для каждой такой зафиксированной пары можно решать описанную выше модифицированную задачу методом динамического программирования при помощи вычисления значений функционала $G(k,l,S)$.

Так как потребуется рассмотреть не более $n_1 \cdot n_2$ модифицированных задач, то для случая $D_2 < D_1$ трудоемкость алгоритма составит $O(n_1^2 \cdot n_2 \cdot \min(n_1,n_2) \cdot (D_2 - B))$.

Список использованной литературы

1. *Танаев, В. С.* Теория расписаний. Многостадийные системы / В. С. Танаев, Ю. Н. Сотсков, В. А. Струсевич. – М.: Наука, 1989.

2. *Ronconi, D. P.* A branch-and-bound algorithm to minimize the makespan in a flowshop with blocking / D. P. Ronconi // Annals of Operations Research. – 2005. – N 138. – P. 53–65.

3. *Танаев, В. С.* Теория расписаний. Групповые технологии / В. С. Танаев, М. Я. Ковалев, Я. М. Шафранский. – Минск: ОИПИ НАН Беларуси, 1998.

Поступила в редакцию 16.01.2016