

## АЛГОРИТМЫ РЕШЕНИЯ ОДНОЙ ЗАДАЧИ ПОСТРОЕНИЯ РАСПИСАНИЯ МИНИМАЛЬНОЙ ДЛИНЫ ДЛЯ ДВУХ МАШИН

*А. А. Романова<sup>1,2</sup>*

<sup>1</sup>Омская юридическая академия,  
ул. Короленко, 12, 644010 Омск, Россия

<sup>2</sup>Омский гос. университет,  
пр. Мира, 55-а, 644077 Омск, Россия

e-mail: anna.a.r@bk.ru

**Аннотация.** Рассматривается NP-трудная задача составления расписания выполнения операций единичной длительности на двух машинах при наличии частичного порядка между операциями с критерием минимизации момента завершения всех операций. Предложен приближённый алгоритм решения задачи с гарантированной оценкой точности. Доказана полиномиальная разрешимость задачи в случае, когда каждая операция, выполняющаяся на первой машине, связана отношениями предшествования ровно с двумя операциями, выполняющимися на второй машине, разработан соответствующий алгоритм. Ил. 9, библиогр. 8.

**Ключевые слова:** кросс-докинг, расписание, частичный порядок, приближённый алгоритм.

### Введение

Появление в последние десятилетия новых систем доставки продукции привело к необходимости решения проблемы их оптимального функционирования. Рассмотрим одну из таких систем, получившую в литературе название кросс-докинг [2]. В системе кросс-докинг процесс приёмки и отгрузки товаров происходит через склад напрямую, без размещения в зоне долговременного хранения, в результате чего продукция доставляется в кратчайшие сроки. Преимуществами такого сквозного складирования являются быстрая доставка продукции заказчикам, сокращение складских площадей и снижение затрат на оплату аренды складов и труд персонала.

При двухэтапном кросс-докинге партии однотипных товаров сначала разгружаются, подвергаются распаковке и переоформлению, а затем из этих товаров формируются заказы конечных потребителей, включающие в себя разные типы товаров. Данные операции выполняют две бригады рабочих. Первая бригада занимается разгрузкой и распаковкой товаров, поэтому одновременно не может разгружаться более одного транспортного средства. Вторая бригада формирует заказы потребителей. Очевидно, что заказ не может быть сформирован, пока не будут разгружены все необходимые для него товары.

Сформулируем данную задачу в терминах теории расписаний в системе flow-shop с двумя машинами, где операция на первой машине — разгрузка и распаковка привезённой продукции, а операция на второй машине — формирование заказа. Работа над очередным заказом не может быть начата, пока все виды продукции из этого заказа не будут разгружены и распакованы. Другими словами, операция на второй машине не может начаться, пока не выполнен некоторый набор операций на первой машине. Необходимо составить расписание обслуживания транспортных средств, доставляющих продукцию на склад, и операций по формированию заказов таким образом, чтобы время завершения формирования всех заказов было минимально.

В данной работе исследуется задача с одинаковыми длительностями выполнения всех операций. Такая задача возникает на практике, когда продукция доставляется однотипными партиями, и на оформление заказов затрачивается одинаковое время. Без ограничения общности будем далее считать, что длительности всех операций равны единице.

Эффективному функционированию системы кросс-докинг посвящено достаточно много работ. Лишь в немногих из них для описания рассматриваемых задач используются термины теории расписаний. Работа [3] — одна из таких. В ней представлен обзор результатов по задачам в системе кросс-докинг, предлагается их классификация. Построению приближённых и эвристических алгоритмов решения задач в системе кросс-докинг посвящён ряд работ (см., например, [5, 7, 8]).

Рассматриваемая задача  $F2|CD|C_{\max}$  NP-трудна в сильном смысле как для произвольных, так и для единичных длительностей операций [4]. Для решения этой задачи в [4] предложен алгоритм ветвей и границ. Следует отметить, что известная полиномиально разрешимая задача Джонсона  $F2||C_{\max}$  [6] является частным случаем этой задачи. Решающее правило для задачи Джонсона используется в некоторых работах для построения приближённого решения. Например, в [4] предложен при-

ближённый алгоритм с гарантированной оценкой точности, который основан на решении вспомогательной задачи  $F2||C_{\max}$ .

В разд. 1 описывается формальная постановка задачи. В разд. 2 предложен приближённый алгоритм с гарантированной оценкой точности, улучшающей оценку из [4]. В разд. 3 выделен полиномиально разрешимый случай задачи, расширяющий случай из [1], приводится соответствующий полиномиальный алгоритм и даётся его обоснование.

## 1. Постановка задачи

Дадим формальную постановку рассматриваемой задачи. Имеются множества операций  $V_1 = \{A_1, \dots, A_n\}$ ,  $V_2 = \{B_1, \dots, B_m\}$  и производственная линия, состоящая из двух машин  $M_1$  и  $M_2$ . Операции  $A_i$  выполняются на машине  $M_1$ , операции  $B_j$  — на машине  $M_2$ . Длительности операций  $A_i$  и  $B_j$  равны единице. Прерывания и одновременное выполнение двух и более операций на одной машине не допускаются. Частичный порядок выполнения операций задан двудольным графом  $G = (V, E)$ , где  $V = V_1 \cup V_2$  — множество вершин,  $E = \{(A_i, B_j) \mid A_i \rightarrow B_j, A_i \in V_1, B_j \in V_2\}$  — множество дуг, задающих отношения предшествования между операциями. Здесь запись  $A_i \rightarrow B_j$  означает, что операция  $B_j$  не может начаться до завершения операции  $A_i$ .

Расписание выполнения операций на машинах определяется заданием времени начала каждой операции. Пусть  $s_i^A$  — время начала операции  $A_i$ , а  $s_j^B$  — время начала операции  $B_j$ . Расписание является допустимым, если к началу выполнения каждой из операций  $B_j$  завершены все операции на первой машине, связанные с ней отношением предшествования.

В рассматриваемой задаче требуется найти допустимое расписание, при котором время завершения всех операций

$$C_{\max} = \max \left\{ \max_{i=1, \dots, n} (s_i^A + 1), \max_{j=1, \dots, m} (s_j^B + 1) \right\}$$

минимально.

Обозначим через  $S_i$  множество операций, связанных отношениями предшествования с операцией  $A_i$ , а через  $T_j$  — множество операций, связанных отношениями предшествования с операцией  $B_j$ . Пусть  $d_i^A = |S_i|$ ,  $d_j^B = |T_j|$ ,  $i = 1, \dots, n$ ,  $j = 1, \dots, m$ .

Так как операции  $A_i$  не имеют предшествующих, их можно выполнять на первой машине без простоев. Поэтому расписание на первой машине можно задать последовательностью выполнения операций. При заданной последовательности выполнения операций на первой машине

расписание выполнения операций на второй машине, при котором время завершения всех операций минимально, можно найти за полиномиальное время (см. разд. 2).

## 2. Приближённый алгоритм

Опишем алгоритм Gr нахождения приближённого решения задачи  $F2|p_j = 1, CD|C_{\max}$ . Идея алгоритма состоит в построении последовательности выполнения операций на первой машине «жадным» образом. Чем больше полустепень исхода вершины первой доли графа, тем раньше имеет смысл выполнить соответствующую операцию.

Упорядочим операции  $A_i$  по невозрастанию величин  $d_i^A$ . При совпадении  $d_i^A$  упорядочим соответствующие операции по невозрастанию величин  $\frac{d_i^A}{\sum_{j \in S_i} d_j^B}$ . Получим последовательность  $\pi$ . Положим  $s_{\pi_i}^A = i - 1$ ,  $i = 1, \dots, n$ .

По последовательности  $\pi$  определяем расписание выполнения операций на второй машине, минимизирующее время завершения всех операций. Вычислим  $r_j = \max \{d_j^B, \max_{i \in T_j} (s_i^A + 1)\}$  — время готовности операции  $B_j$  к выполнению. Действительно, операция  $B_j$  не может начаться раньше завершения всех предшествующих ей операций на первой машине. Далее, сортируем последовательность операций на второй машине по неубыванию  $r_j$  и в данном порядке выполняем их, соблюдая условие  $s_j^B \geq r_j$ . Данный алгоритм является полиномиальным с трудоёмкостью  $O(n \log_2 n + nm)$  операций.

Обозначим через  $q_A$  наименьший номер, при котором выполняется неравенство

$$\sum_{i=1}^{q_A} d_{\pi_i}^A > \sum_{i=1}^n d_{\pi_i}^A - m. \quad (1)$$

В следующей теореме получена оценка точности предложенного алгоритма.

**Теорема 1.** Пусть  $C_{\max}^*$  — длина оптимального расписания,  $C_{\max}^{\text{Gr}}$  — длина расписания, полученного алгоритмом Gr. Тогда

$$\frac{C_{\max}^{\text{Gr}}}{C_{\max}^*} \leq \frac{\max\{q_A + m, n\}}{\max\{n + d_{\min}^A, m + d_{\min}^B\}},$$

где  $d_{\min}^A = \min_{i=1, \dots, n} d_i^A$ ,  $d_{\min}^B = \min_{j=1, \dots, m} d_j^B$ .

ДОКАЗАТЕЛЬСТВО. Очевидно, что  $\max\{n + d_{\min}^A, m + d_{\min}^B\}$  — нижняя граница длины расписания. Построим верхнюю границу длины расписания, которое может быть получено алгоритмом Gr. Рассмотрим вектор  $d^B = (d_1^B, \dots, d_m^B)$ . Заметим, что операция  $A_i$  вносит вклад, равный единице, в значения  $d_j^B$  для всех  $j \in S_i$ . После выполнения каждой операции на первой машине пересчитываем вектор  $d^B$ , вычитая по единице из соответствующих  $d_j^B$ . При этом операция  $B_j$  может поступать в обработку только в случае  $d_j^B = 0$ . Построим допустимое расписание, соответствующее построенной алгоритмом последовательности  $\pi$  выполнения операций на первой машине. Выполним операции  $A_{\pi_1}, A_{\pi_2}, \dots, A_{\pi_q}$  на первой машине, не выполняя операций на второй машине. После этого в силу неравенства (1) сумма координат вектора  $d^B$  станет меньше числа  $m$  операций на второй машине. Это значит, что после выполнения  $q_A$  операций на первой машине по крайней мере одна операция  $B_z$  на второй машине доступна к выполнению. Докажем, что, начиная с периода времени  $q_A + 1$ , операции на второй машине можно выполнять без простоев. В соответствии с алгоритмом операции на первой машине выполняются по невозрастанию полустепеней исхода соответствующих вершин графа. Пусть  $v$  — наибольшая полустепень исхода оставшихся вершин первой доли графа. Очевидно, что  $v \geq 1$ . Во время выполнения соответствующей операции  $A_{\pi_{q_A+1}}$  можно выполнить операцию  $B_z$  на второй машине. Сумма координат вектора  $d^B$  после этого уменьшится на  $v$ , т. е. станет меньше чем  $m - v$ . При этом на второй машине останется  $m - 1$  операций для выполнения. Это значит, что опять по крайней мере одна операция на второй машине доступна для выполнения во время выполнения  $(q_A + 2)$ -й операции на первой машине. Такая же ситуация повторяется и далее. После очередной операции на первой машине по крайней мере одна операция на второй машине становится доступна для выполнения. Это продолжается до тех пор, пока вектор  $d^B$  не станет нулевым. Тогда оставшиеся операции на обеих машинах могут выполняться без простоев. Длина соответствующего расписания будет равна  $\max\{q_A + m, n\}$ . Таким образом, длина расписания, полученного алгоритмом Gr, не превышает  $\max\{q_A + m, n\}$ . Теорема 1 доказана.

Заметим, что допустимое расписание можно также получить, применив алгоритм Gr к так называемой обратной задаче, которая отличается от исходной противоположным направлением дуг графа предшествования. Полученное расписание для обратной задачи преобразуется в допустимое расписание для исходной, если запустить его в противоположном направлении, начиная с момента завершения всех операций. Применяя

алгоритм Gr как для исходной задачи, так и для обратной, можно улучшить его оценку точности до

$$\frac{\min\{\max\{q_A + m, n\}, \max\{q_B + n, m\}\}}{\max\{n + d_{\min}^A, m + d_{\min}^B\}},$$

где  $q_B$  — наименьший номер, при котором

$$\sum_{j=1}^{q_B} d_{\pi_j}^B > \sum_{j=1}^m d_{\pi_j}^B - n,$$

и  $\pi$  — последовательность выполнения операций на машине  $M_2$  во вспомогательной обратной задаче.

В [4] предложен приближённый алгоритм  $F2R$  для рассматриваемой задачи в случае произвольных длительностей операций с оценкой точности  $(2 - \frac{1}{\min\{d_{\max}^A, d_{\max}^B\}})$ , где  $d_{\max}^A = \max_{i=1, \dots, n} d_i^A$ ,  $d_{\max}^B = \max_{j=1, \dots, m} d_j^B$ . Заметим, что при применении данного алгоритма к задачам с единичными длительностями оценку точности можно улучшить. В ходе работы алгоритма  $F2R$  сначала строится и решается вспомогательная задача  $F2||C_{\max}$  с  $n$  работами, если  $m - m/d_{\max}^B \geq n - n/d_{\max}^A$ , и  $m$  работами — в противном случае. Далее по полученной последовательности операций на первой либо на второй машине достраивается допустимое решение для исходной задачи. Таким образом, трудоёмкость данного алгоритма равна  $O(n \log_2 n + nm)$  либо  $O(m \log_2 m + nm)$  операций. В любом случае трудоёмкость алгоритма  $F2R$  по порядку такая же, как трудоёмкость предлагаемого в работе алгоритма Gr.

Легко видеть, что в случае единичных длительностей целевая функция решения вспомогательной задачи не превышает  $\max\{m+1, n+1\}$ . По аналогии с выкладками из [4] оценим значение  $C_{\max}^{F2R}$  целевой функции приближённого решения, полученного алгоритмом  $F2R$ :

$$C_{\max}^{F2R} \leq \max\{m+1, n+1\} + \min\{m - m/d_{\max}^B, n - n/d_{\max}^A\}.$$

Покажем, что целевую функцию решения, полученного алгоритмом Gr, можно оценить сверху меньшей величиной. Заметим, что при  $d_{\max}^A = 1$  или  $d_{\max}^B = 1$  как алгоритм Gr, так и алгоритм  $F2R$  находят оптимальное решение. Исключим эти случаи из рассмотрения и далее будем считать, что  $d_{\max}^A \geq 2$  и  $d_{\max}^B \geq 2$ .

Заметим, что  $q_A \leq n - \lfloor m/d_{\max}^A \rfloor$ ,  $q_B \leq m - \lfloor n/d_{\max}^B \rfloor$ . Оценим значение целевой функции решения, полученного алгоритмом Gr:

$$\begin{aligned}
C_{\max}^{\text{Gr}} &\leq \min\{\max\{q_A + m, n\}, \max\{q_B + n, m\}\} \\
&\leq \min\{\max\{n - \lfloor m/d_{\max}^A \rfloor + m, n\}, \max\{m - \lfloor n/d_{\max}^B \rfloor + n, m\}\} \\
&\leq m+n - \max\{\lfloor m/d_{\max}^A \rfloor, \lfloor n/d_{\max}^B \rfloor\} < m+n+1 - \max\{m/d_{\max}^A, n/d_{\max}^B\}.
\end{aligned}$$

При  $n \geq m$  выполняется

$$\begin{aligned}
m+n+1 - \max\{m/d_{\max}^A, n/d_{\max}^B\} &\leq m+n+1 - n/d_{\max}^B \\
&\leq m+n+1 - m/d_{\max}^B \leq \max\{m+1, n+1\} + m - m/d_{\max}^B,
\end{aligned}$$

а также

$$\begin{aligned}
m+n+1 - \max\{m/d_{\max}^A, n/d_{\max}^B\} &\leq m+n+1 - m/d_{\max}^A \\
&\leq n+1 + m(1 - 1/d_{\max}^A) \leq \max\{m+1, n+1\} + n - n/d_{\max}^A.
\end{aligned}$$

При  $n < m$  выполняется

$$\begin{aligned}
m+n+1 - \max\{m/d_{\max}^A, n/d_{\max}^B\} &\leq m+n+1 - n/d_{\max}^B \\
&\leq m+1 + n(1 - 1/d_{\max}^B) < \max\{m+1, n+1\} + m - m/d_{\max}^B,
\end{aligned}$$

а также

$$\begin{aligned}
m+n+1 - \max\{m/d_{\max}^A, n/d_{\max}^B\} &\leq m+n+1 - m/d_{\max}^A \\
&< m+1 + n - n/d_{\max}^A \leq \max\{m+1, n+1\} + n - n/d_{\max}^A.
\end{aligned}$$

Во всех случаях

$$\begin{aligned}
&\min\{\max\{q_A + m, n\}, \max\{q_B + n, m\}\} \\
&< \max\{m+1, n+1\} + \min\{m - m/d_{\max}^B, n - n/d_{\max}^A\}.
\end{aligned}$$

Таким образом, верхняя оценка длины расписания, полученного алгоритмом Gr, меньше оценки длины расписания, полученного F2R, значит, оценка точности алгоритма Gr лучше оценки точности алгоритма F2R, применённого для задачи с единичными длительностями.

Заметим, что чем меньше величины  $d_{\max}^A$  и  $d_{\max}^B$ , т. е. чем «ближе» задача к F2||C<sub>max</sub>, тем лучше оценка точности алгоритма F2R. Несмотря на то, что величина  $m+n+1 - \max\{m/d_{\max}^A, n/d_{\max}^B\}$  незначительно меньше величины  $\max\{m+1, n+1\} + \min\{m - m/d_{\max}^B, n - n/d_{\max}^A\}$ , оценка алгоритма Gr при больших значениях  $d_{\max}^A$  и  $d_{\max}^B$  может быть существенно лучше оценки алгоритма F2R. Это объясняется тем, что величины  $q_A$  и  $q_B$  при сравнении оценок были оценены достаточно грубо.

### 3. Полиномиально разрешимый случай

Рассмотрим класс примеров задачи  $F2|p_j = 1, CD|C_{\max}$ , в которых вершины первой доли  $V_1$  графа  $G$  имеют степень 2. В дальнейшем данный класс будем обозначать через  $D_2$ . В [1] доказана полиномиальная разрешимость класса  $D_2$  в случае связного графа  $G$  и построен соответствующий алгоритм  $PD_2$ . В данной работе показано, что алгоритм  $PD_2$  находит оптимальное решение для всех примеров из класса  $D_2$ . Приведём этот алгоритм.

#### Алгоритм $PD_2$

Пока  $V_2 \neq \emptyset$  выполнять

ШАГ 1. Упорядочим текущее множество  $V_2$  операций на второй машине по неубыванию  $d_j^B$ :  $d_{l_1}^B \leq d_{l_2}^B \leq \dots \leq d_{l_k}^B$ , где  $k = |V_2|$ . Операции  $B_{l_j}$ ,  $j = 1, \dots, r$ , для которых  $T_{l_j} = \emptyset$ , выполним в произвольном порядке на машине  $M_2$  в момент её освобождения и соответствующие вершины удалим из множества  $V_2$ .

ШАГ 2. В момент освобождения первой машины выполним  $d_{l_{r+1}}^B$  операций множества  $T_{l_{r+1}}$ . Удалим все дуги, выходящие из этих вершин, вместе с вершинами. Выполним операцию  $B_{l_{r+1}}$  на второй машине с соблюдением условий допустимости и удалим соответствующую вершину из множества  $V_2$ . Обновим множества  $T_j$  и величины  $d_j^B$  для оставшихся вершин  $B_j \in V_2$ .

КОНЕЦ ЦИКЛА.

Алгоритм  $PD_2$  полиномиальный с трудоёмкостью  $O(m^2 \log_2 m + mn)$  операций.

Разобьём расписание, полученное в результате работы алгоритма, на блоки. Пусть  $\min_{j=1, \dots, m} d_j^B = t$ . Значит, первая выбранная вершина будет степени  $t$ . Эта вершина и вершины, связанные с ней отношениями предшествования, дают начало блоку  $BL_t$ . Последующие включаемые в расписание операции будут отнесены к этому же блоку до тех пор, пока на шаге 2 величина  $d_{l_{r+1}}^B$  не станет больше  $t$ . Если в какой-то момент времени на шаге 1 будет  $\min_{j=1, \dots, m} d_j^B = g > t$ , то включаемые в расписание на шаге 2 операции дадут начало блоку  $BL_g$ .

Рассмотрим блок  $BL_i$ . Множество операций блока  $BL_i$  на первой машине, выполняющихся до всех операций второй машины этого же блока, будем называть *отступом* блока  $BL_i$ , а время их выполнения (численно равное их количеству) — *длиной отступа* блока  $BL_i$ . Очевидно, что



длина отступа в каждом таком блоке равна  $i$ . Множество операций блока  $BL_i$ , выполняющихся на второй машине, которые не могут начаться раньше, чем завершится последняя операция этого блока на первой машине, будем называть *выступом* блока  $BL_i$ , время их выполнения — *длиной выступа*. В силу структуры решаемого класса задач длина выступа всегда равна 2. В общем виде блок  $BL_i$  можно изобразить, как показано на рис. 1.

Если на второй машине есть простои, то по построению блока  $BL_i$  продолжительность каждого из них не превышает  $i$ . Заметим, что блок  $BL_0$  состоит только из операций, выполняющихся на второй машине.

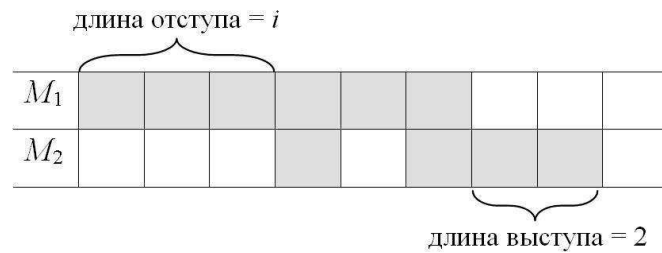


Рис. 1. Структура блока  $BL_i$

Расписание, получаемое алгоритмом, можно представить в виде состыкованных блоков  $BL_{v_j}$ ,  $j = 1, \dots, h$ , при этом  $v_1 < v_2 < \dots < v_h$ . Под стыковкой понимаем «склеивание» блоков по операциям на первой машине так, чтобы они выполнялись без простоев. Блок может менять конфигурацию при стыковке. Операции на первой машине могут сдвинуться влево из-за требования выполнения их без простоев. Операции на второй машине могут сдвинуться вправо из-за её занятости в нужный момент (особенно при наличии блока  $BL_0$ ).

Для иллюстрации работы алгоритма и введённых понятий и обозначений приведём пример. Граф  $G$  представлен на рис. 2.

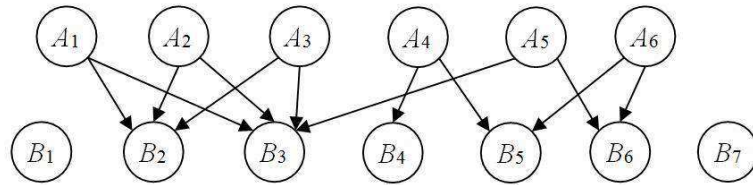


Рис. 2. Пример

На шаге 1 сортируем вершины второй доли графа по неубыванию числа входящих дуг. Получим последовательность  $(B_1, B_7, B_4, B_5, B_6,$

$B_2, B_3$ ) и соответствующий вектор полустепеней захода  $d^B = (0, 0, 1, 2, 2, 3, 4)$ . Включаем в расписание операции  $B_1$  и  $B_7$ , не связанные отношениями предшествования с операциями на первой машине, и удаляем соответствующие вершины из графа. Эти операции образуют блок  $BL_0$ . Так как степень следующей в списке операции  $B_4$  равна 1, на шаге 2 операции  $A_4, B_4$  дают начало блоку  $BL_1$ . Удаляем соответствующие вершины из графа вместе с дугами, выходящими из вершины  $A_4$ . Переходим на шаг 1. Сортируем оставшиеся вершины второй доли по неубыванию числа входящих дуг. Получим последовательность  $(B_5, B_6, B_2, B_3)$  оставшихся вершин второй доли и соответствующий вектор степеней  $d^B = (1, 2, 3, 4)$ . Как видно, минимальная степень среди оставшихся вершин равна 1, поэтому построение блока  $BL_1$  продолжается. На шаге 2 в расписание включаются операции  $A_6, B_5$  и удаляются дуги, выходящие из  $A_6$ . Переходя на шаг 1, заново сортируем оставшиеся вершины второй доли; получаем последовательность  $(B_6, B_2, B_3)$  и вектор  $d^B = (1, 3, 4)$ . Блок  $BL_1$  продолжается операциями  $A_5$  и  $B_6$ . Остаются вершины  $B_2, B_3$  со степенями, равными 3. Построение блока  $BL_3$  начинается вершинами  $A_1, A_2, A_3$  и  $B_4$ . Удаляем соответствующие вершины из графа вместе с дугами, выходящими из вершин  $A_1, A_2, A_3$ . Во второй доле осталась вершина  $B_3$  нулевой степени, которая завершает построение блока  $BL_3$ . Описанные блоки представлены на рис. 3.

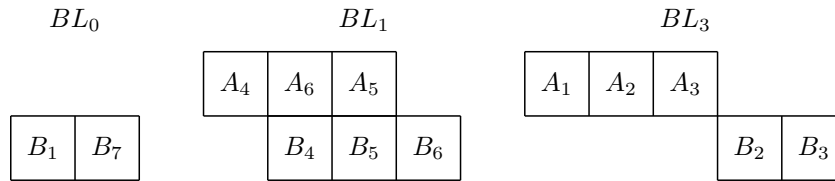


Рис. 3. Блоки

Расписание, полученное в результате стыковки данных блоков, изображено на рис. 4.

$M_1$	$A_4$	$A_6$	$A_5$	$A_1$	$A_2$	$A_3$			
$M_2$	$B_1$	$B_7$	$B_4$	$B_5$	$B_6$		$B_2$	$B_3$	

Рис. 4. Оптимальное расписание

Докажем, что описанный алгоритм находит оптимальное решение в рассматриваемом классе задач. Начнём со вспомогательной леммы.

**Лемма 1.** В блоке  $BL_k$  при  $k \geq 2$  длина выступа равна 1 либо 2.

**Доказательство.** Рассмотрим процесс образования блока  $BL_k$ , где  $k \geq 2$ , начиная с рассмотрения некоторой вершины  $B_{i_1}$ , полустепень захода которой равна  $k$ . Отметим, что в начале формирования блока полустепени захода всех остальных вершин второй доли не меньше  $k$ . Удаляя  $k$  вершин первой доли графа, связанных с вершиной  $B_{i_1}$ , вместе с инцидентными им  $2k$  дугами, уменьшаем на  $2k$  сумму степеней вершин второй доли. При этом на  $k$  уменьшается степень вершины  $B_{i_1}$ , значит, на остальные вершины приходится суммарное уменьшение, равное  $k$ . После удаления дуг и вершины  $B_{i_1}$  не может образоваться более двух вершин, полустепень захода которых равна 1, так как для появления таких вершин необходимо удалить не менее  $k - 1$  инцидентных им дуг. По той же причине не может образоваться более одной висячей вершины. На рис. 5 изображена текущая конфигурация блока после применения шага 2.

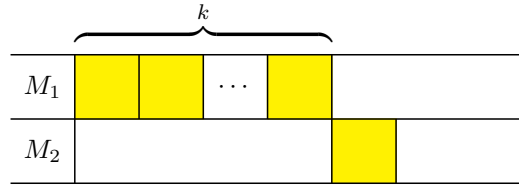


Рис. 5. Начало блока  $BL_k$

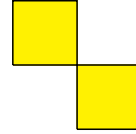


Рис. 6. Фрагмент блока  $BL_k$

При этом возможны следующие варианты для оставшихся вершин второй доли:

- А) минимальная степень вершин равна  $t$ , где  $t \in \{2, \dots, k\}$ ,
- Б) имеются две вершины степени 1, остальные — большей степени (при их наличии),
- В) имеется одна вершина степени 1, остальные — большей степени (при их наличии),
- Г) имеется одна висячая вершина, остальные — большей степени (при их наличии),
- Д) множество вершин пусто,
- Е) минимальная степень вершин больше  $k$ .

Заметим, что при возникновении варианта Е процесс построения блока завершен, текущее состояние блока удовлетворяет условиям леммы.

Если появился вариант А, то на следующей итерации к блоку добавится фрагмент, аналогичный изображённому на рис. 5 (выступ не меньше 2 и не превышает  $k$ ). В дальнейшем опять возможны варианты А–Е.

При появлении варианта Б к блоку добавляется фрагмент, изображённый на рис. 6. Если эти две вершины инцидентны разным вершинам первой доли, то приходим к варианту В; если они связаны с одной и той же вершиной, то — к варианту Г.

Если появился вариант В, то к блоку также добавляется фрагмент, изображённый на рис. 6. В дальнейшем, очевидно, опять приходим к варианту А, Д или Е.

При появлении варианта Г добавляется соответствующая операция на вторую машину, выступ блока становится равным 2. В дальнейшем возможны варианты А, Д или Е.

Заметим, что при появлении варианта А отступ очередного фрагмента не меньше 2, поэтому при простой стыковке фрагментов не образуется простоя на первой машине, а это значит, что выступ опять станет равным 1. Всевозможные переходы между вариантами изображены на рис. 7.

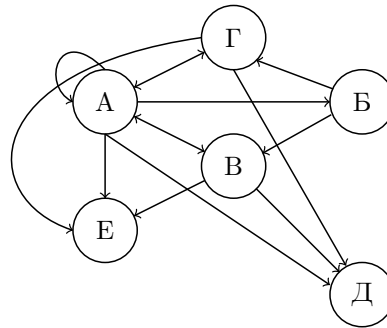


Рис. 7. Процесс построения блока

Очевидно, что процесс построения блока конечен и завершится вариантом Д или Е. Фрагменты, добавляемые к блоку при возникновении вариантов А–Г, не образуют простоя на первой машине, так как отступ любого добавляемого фрагмента не меньше двух, а длина выступа, как видно, всегда получается не больше двух. Так как длина отступа в фрагментах всегда не меньше 2, а длина выступа не превышает двух, стыковкой получим блок с длиной выступа, не превышающей 2. Лемма 1 доказана.

Переходим к доказательству основного результата раздела.

**Теорема 2.** В задаче  $F2|p_j = 1, CD|C_{\max}$ , в которой каждая операция, выполняющаяся на первой машине, связана отношениями предшествования ровно с двумя операциями, выполняемыми на второй машине, алгоритм  $PD_2$  находит оптимальное расписание.

ДОКАЗАТЕЛЬСТВО. При наличии висячих вершин во второй доле нижняя граница длины расписания равна  $\max\{n + 2, m\}$  (т. е. при наличии блока  $BL_0$ ) и равна  $\max\{n + 2, m + 1\}$  при отсутствии висячих вершин во второй доле.

В силу леммы 1 у блоков  $BL_t$ ,  $t \geq 2$ , выступ равен 1 или 2. Будем называть такие блоки *старшими*. У последнего блока выступ обязательно равен 2 (так как последняя операция на первой машине связана ровно с двумя операциями на второй машине). Если общее число операций на первой машине в старших блоках равно  $w$ , то длина фрагмента, образованного их стыковкой, равна  $w + 2$ . Действительно, длина отступа каждого следующего блока больше, чем длина выступа предыдущего, поэтому операции на первой машине выполняются без простоев; выступ последнего блока равен 2.

Таким образом, если в расписании отсутствуют блоки  $BL_0$  и  $BL_1$ , алгоритм найдёт расписание длины  $n + 2$ . Заметим, что в этом случае в силу структуры задачи количество операций, выполняющихся на первой машине, не меньше количества операций, выполняющихся на второй машине, т. е.  $n \geq m$ , поэтому нижняя граница длины расписания равна  $n + 2$ . Значит, найденное расписание оптимально.

Пусть в расписании имеется блок  $BL_1$ , но нет висячих вершин во второй доле (т. е. блок  $BL_0$  отсутствует). Длина отступа блока равна 1. Рассмотрим процесс построения блока по аналогии со старшими блоками. Пусть вершина  $B_v$  имеет степень 1. Блок начинается с фрагмента, изображённого на рис. 6, где на первой машине выполняется операция, связанная с  $B_v$ , на второй машине — сама операция  $B_v$ . После удаления двух дуг в соответствии с алгоритмом (одной — к вершине  $B_v$ ) может образоваться либо одна висячая вершина во второй доле (вариант А: в этом случае длина выступа станет равной 2), либо — вершина полустепени захода 1 (вариант Б: тогда процесс образования блока продолжится и стыковка следующего фрагмента с текущим произойдёт без простоев на каждой из машин), либо полустепень захода остальных вершин станет больше 1 (вариант В: тогда процесс построения блока закончен). После варианта А переходим либо в вариант Б, либо к варианту В. При этом длина выступа увеличивается на 1. При варианте Б длина выступа не меняется, так как добавляется одна операция на машину  $M_1$  и одна — на машину  $M_2$ . В любом случае можно добиться выполнения операций блока на машинах без простоев между операциями (для машины  $M_2$  — без учёта простоя в первый момент времени), т. е. блок  $BL_1$  имеет вид, как показано на рис. 8.

Соединяем данный блок со старшими блоками. Выступ первого блока может быть больше отступа первого старшего блока. В этом случае операции старших блоков на первой машине сдвигаем влево так, чтобы они выполнялись без простоев. С помощью выступа первого блока сдвигаем вправо некоторые операции второй машины, избавляясь от некоторых простоев в старших блоках. Если  $m + 1 \geq n + 2$ , то от всех простоев на второй машине, кроме первого момента, удастся избавиться, и длина расписания будет равна  $m + 1$ . Если  $n + 2 > m + 1$ , то  $n + 1 - m$  простоев всё же останется. Результирующее расписание, начиная с некоторой операции, будет совпадать с фрагментом, полученным стыковкой старших блоков, поэтому длина выступа будет такой же, как в последнем старшем блоке, т. е. равна 2. Таким образом, длина расписания, найденного алгоритмом, будет равна  $n + 2$ . В обоих случаях длина расписания совпадает с нижней границей длины расписания при отсутствии висячих вершин, а значит, алгоритм строит оптимальное расписание.

$M_1$									
$M_2$							...		

Рис. 8. Блок  $BL_1$ 

Наконец, если в расписании присутствует блок  $BL_0$  (есть висячие вершины во второй доле графа), то при стыковке блоков  $BL_0$  и  $BL_1$  образуется фрагмент, изображённый на рис. 9.

$M_1$									
$M_2$							...		

Рис. 9. Стыковка блоков  $BL_0$  и  $BL_1$ 

Дальнейшие рассуждения аналогичны уже приведённым с отличием в том, что на второй машине в первый момент нет простоя.

Если из младших блоков присутствует только блок  $BL_0$ , то операции этого блока сдвигают операции старших блоков, выполняющиеся на второй машине. Далее рассуждения проводятся по аналогии. Алгоритм построит расписание длины  $\max\{n + 2, m\}$ , что совпадает с нижней границей при наличии висячих вершин. Теорема 2 доказана.

Итак, задачи класса  $D_2$  полиномиально разрешимы. Однако вопрос о расширении данного класса, например, на задачи, в которых степени

вершин первой доли графа не превосходят 2, пока остаётся открытым, так как алгоритм  $PD_2$  в этом случае, к сожалению, не всегда даёт оптимальное решение. При решении задач, в которых степени вершин первой доли графа равны 3, алгоритм  $PD_2$  также не всегда даёт оптимальное решение.

Исследование сложности различных классов задач как с единичными, так и с произвольными длительностями операций, является одним из перспективных направлений дальнейшей работы.

### ЛИТЕРАТУРА

1. Романова А. А., Чередова И. Б. Задача обработки деталей на двух станках при наличии частичного порядка между операциями // Тр. V Междунар. азиатской школы-семинара «Проблемы оптимизации сложных систем» (Бишкек, Кыргызстан, 12–22 августа, 2009 г.). Новосибирск: Ин-т вычисл. математики и мат. геофизики СО РАН, 2009. С. 119–124.
2. Apte U. M., Viswanathan S. Effective cross docking for improving distribution efficiencies // Int. J. Logist. Res. Appl. 2000. Vol. 3, No. 3. P. 291–302.
3. Boysen N., Fliedner M. Cross dock scheduling: Classification, literature review and research agenda // Omega. 2010. Vol. 38, No. 6. P. 413–422.
4. Chen F., Lee Ch.-Yee. Minimizing the makespan in a two-machine cross-docking flow-shop problem // Eur. J. Oper. Res. 2009. Vol. 193, No. 1. P. 59–72.
5. Chen F., Song K. Minimizing makespan in two-stage hybrid cross docking scheduling problem // Comput. Oper. Res. 2009. Vol. 36, No. 6. P. 2066–2073.
6. Johnson S. M. Optimal two- and three-stage production schedules with set-up times included // Naval Res. Logist. Quart. 1954. Vol. 1, No. 1. P. 61–68.
7. Prot D., Rapine C. Approximations for the two-machine cross-docking flow shop problem // Discrete Appl. Math. 2013. Vol. 161, No. 13–14. P. 2107–2119.
8. Vahdani B., Zandieh M. Scheduling trucks in cross-docking systems: Robust meta-heuristics // Comput. Ind. Eng. 2010. Vol. 58, No. 1. P. 12–24.

Романова Анна Анатольевна

Статья поступила  
17 марта 2015 г.

Исправленный вариант —  
6 мая 2015 г.

EFFECTIVE ALGORITHMS FOR ONE SCHEDULING PROBLEM  
ON TWO MACHINES WITH MAKESPAN MINIMIZATIONA. A. Romanova<sup>1,2</sup><sup>1</sup>Omsk Law Academy,

12 Korolenko St., 644010 Omsk, Russia

<sup>2</sup>Omsk State University,

55-a Mir Ave., 644077 Omsk, Russia

e-mail: anna.a.r@bk.ru

**Abstract.** We consider an NP-hard problem of scheduling a set of jobs of equal processing time on two machines, given a partial precedence order between the jobs, with an objective to minimize the makespan. An approximation algorithm with performance guarantee is proposed for this problem. Polynomial solvability of the problem is proved in the case when each job on the first machine is in precedence relation with two jobs on the second machine. Ill. 9, bibliogr. 8.

**Keywords:** cross-docking problem, schedule, partial order, approximation algorithm.

## REFERENCES

1. A. A. Romanova and I. B. Cheredova, The scheduling problem for two machines with partial precedence order, in *Tr. V Mezhdunarodnoi aziatskoi shkoly-seminara "Problemy optimizatsii slozhnykh sistem"* (Proc. V Int. Asian School-Seminar "Optimization Problems of Complex Systems"), Bishkek, Kyrgyzstan, Aug. 12–22, 2009, pp. 119–124, Inst. Vychisl. Mat. Mat. Geofiz., Novosibirsk, 2009.
2. U. M. Apte and S. Viswanathan, Effective cross docking for improving distribution efficiencies, *Int. J. Logist., Res. Appl.*, **3**, No. 3, 291–302, 2000.
3. N. Boysen and M. Fliedner, Cross dock scheduling: Classification, literature review and research agenda, *Omega*, **38**, No. 6, 413–422, 2010.
4. F. Chen and Ch.-Yee Lee, Minimizing the makespan in a two-machine cross-docking flow shop problem, *Eur. J. Oper. Res.*, **193**, No. 1, 59–72, 2009.
5. F. Chen and K. Song, Minimizing makespan in two-stage hybrid cross docking scheduling problem, *Comput. Oper. Res.*, **36**, No. 6, 2066–2073, 2009.



6. **S. M. Johnson**, Optimal two- and three-stage production schedules with setup times included, *Nav. Res. Logist. Q.*, **1**, No. 1, 61–68, 1954.
7. **D. Prot** and **C. Rapine**, Approximations for the two-machine cross-docking flow shop problem, *Discrete Appl. Math.*, **161**, No. 13–14, 2107–2119, 2013.
8. **B. Vahdani** and **M. Zandieh**, Scheduling trucks in cross-docking systems: Robust meta-heuristics, *Comput. Ind. Eng.*, **58**, No. 1, 12–24, 2010.

*Anna A. Romanova*

Received

17 March 2015

Revised

6 May 2015