

СЛОЖНОСТЬ ЗАДАЧИ ПОСТРОЕНИЯ ЦИКЛИЧЕСКОГО  
РАСПИСАНИЯ ОБРАБОТКИ ПАРТИИ ИДЕНТИЧНЫХ  
ДЕТАЛЕЙ С ТЕХНОЛОГИЧЕСКИМИ ОГРАНИЧЕНИЯМИ \*)

*А. А. Романова<sup>1,a</sup>, В. В. Сервах<sup>2,b</sup>*

<sup>1</sup> Омский гос. университет им. Ф. М. Достоевского,  
пр. Мира, 55а, 644077 Омск, Россия

<sup>2</sup> Омский филиал Института математики им. С. Л. Соболева,  
ул. Певцова, 13, 644099 Омск, Россия

E-mail: <sup>a</sup> [anna.a.r@bk.ru](mailto:anna.a.r@bk.ru), <sup>b</sup> [svv\\_usa@rambler.ru](mailto:svv_usa@rambler.ru)

**Аннотация.** Рассматривается задача построения циклического расписания обработки идентичных деталей с минимальным временем цикла при наличии запретов на простой между некоторыми парами последовательно выполняемых операций. Исследована сложность задачи и ряда её частных случаев. Доказано, что в общем случае задача является NP-трудной в сильном смысле. Выделены эффективно разрешимые случаи и описаны соответствующие алгоритмы. В случае обработки детали без простоев между операциями доказана полиномиальная разрешимость задачи и предложено два полиномиальных алгоритма. Разработан алгоритм решения задачи в общем случае, который является псевдополиномиальным, если число пар последовательных операций, между которыми разрешены простои, ограничено константой. С другой стороны, показано, что в случае одного запрета задача полиномиально разрешима, а при двух запретах уже NP-трудна. Ил. 4, библиогр. 14.

**Ключевые слова:** теория расписаний, циклическое расписание, идентичные детали, теория сложности, полиномиальный алгоритм, псевдополиномиальный алгоритм.

### Введение

На многих предприятиях производство представляет собой сложный технологический процесс, для управления которым требуется решать большое число задач эффективного распределения ресурсов и построения расписаний. При обработке идентичных деталей на производстве

---

\*) Работа В. В. Сервах выполнена при финансовой поддержке Программы фундаментальных научных исследований СО РАН № I.5.1 (проект № 0314–2019–0019).

традиционно используется конвейер, на котором выпуск деталей носит циклический характер. В последнее время получили распространение высокотехнологичные производственные линии. Это привело к новым постановкам задач обработки идентичных деталей со сложным технологическим маршрутом и другими ограничениями.

В циклических расписаниях выполнение одних и тех же операций любых двух последовательных деталей начинается через равные промежутки времени. Это время называется временем цикла. Естественным критерием построения циклического расписания является критерий минимизации времени цикла, обеспечивающий в пределе максимальную производительность линии. Задача с данным критерием относится к числу полиномиально разрешимых.

Сложность задач построения циклических расписаний с различными технологическими ограничениями, в том числе при многократном возвращении детали на некоторые машины, исследовалась в [1–7]. В нашей работе рассматриваются задачи обработки идентичных деталей с произвольным технологическим маршрутом (Job Shop). Большинство таких задач NP-трудны в сильном смысле [4, 7], поэтому целесообразным является построение приближённых и эвристических алгоритмов их решения. Такие алгоритмы предлагаются в [2, 8–10]. С другой стороны, в [2, 4, 7] разработаны точные алгоритмы решения задач, основанные на методе ветвей и границ и методе динамического программирования.

На производстве часто возникают ситуации, когда после завершения какой-то операции необходимо сразу начать выполнение следующей операции технологического маршрута, т. е. запрещаются простои между некоторыми парами последовательных операций. Такая ситуация возникает, например, при химической обработке деталей, когда после некоторых операций (хромирование и т. п.) следующая операция должна проводиться немедленно, а после некоторых операций (сушка, промывка и т. п.) таких условий нет. При наличии подобных ограничений, называемых в литературе no-wait, как правило, длина цикла возрастает. Данная работа посвящена исследованию сложности данной задачи и различных её подслучаев.

В разд. 1 приводится формальная постановка задачи. В разд. 2 показана её сильная NP-трудность. В разд. 3 построен полиномиальный алгоритм решения задачи в случае, когда при обработке каждой детали простои между операциями запрещены. В разд. 4 предлагается псевдополиномиальный алгоритм в случае, когда простои разрешены лишь в фиксированном количестве мест технологического маршрута. В разд. 5 исследуется задача с малым числом запретов. Доказано, что уже при двух запретах задача NP-трудна.

### 1. Постановка задачи

Рассмотрим задачу обработки большой партии идентичных деталей на производственной линии, состоящей из  $m$  различных машин. Обработка детали состоит из  $n$  последовательно выполняемых операций  $O_1, O_2, \dots, O_n$ . Операция  $O_j$  выполняется на машине  $m_j$  в течение  $p_j \in Z^+$  единиц времени без прерываний. Допускается многократное поступление детали на машину. На каждой машине не может выполняться более одной операции одновременно. Технологически запрещены простои между некоторыми парами последовательно выполняемых операций. Обозначим через  $Z = \{(O_j, O_{j+1}) \mid j \in N_Z\}$  множество пар операций, на которые наложено указанное ограничение. Здесь  $N_Z \subseteq \{1, 2, \dots, n-1\}$  — множество номеров операций, стоящих первыми в запретных парах.

Пусть  $t(j, k)$  — время начала выполнения операции  $O_j$  детали  $k$ . *Расписание* определяется временем начала выполнения каждой операции  $O_j$  для каждой детали  $k$ ,  $j = 1, \dots, n$ ,  $k \in Z$ . Расписание *допустимо*, если на каждой машине одновременно выполняется не более одной операции, соблюдаются технологический порядок операций и ограничения на простои.

Расписание называется *циклическим*, если для любых  $j = 1, \dots, n$  и  $k \in Z$  выполняется  $t(j, k) = t(j, k-1) + C$ , где  $C$  — *время цикла*, или *циклическое время*. Циклическое расписание определяется заданием времени цикла  $C$  и времени  $t_j$  начала выполнения операции  $O_j$  одной из деталей для всех  $j = 1, \dots, n$ . Необходимо найти допустимое циклическое расписание с минимальным временем цикла.

Через промежуток времени, равный времени цикла, получается готовая деталь. Чем меньше время цикла, тем больше будет изготовлено деталей на производстве в одну единицу времени. Поэтому минимизация времени цикла в случае достаточно большой партии деталей обеспечивает максимальную производительность линии.

Задача построения циклического расписания с критерием минимизации времени цикла без запретов на простои между операциями полиномиально разрешима. Минимальное время цикла  $C_{\min}$  при этом равно суммарному времени выполнения всех операций одной детали на самой загруженной машине. Приведём алгоритм решения этой задачи, так как он будет использоваться в дальнейшем. Без ограничения общности первую операцию начинаем в момент начала цикла. Если вторая операция использует другую машину, то её выполняем тоже в начале цикла, но уже следующего, а в текущем цикле будет выполняться вторая операция предшествующей детали. Каждую следующую операцию начинаем выполнять в момент, когда освобождается нужная машина. Если при этом

нарушается технологический порядок операций одной детали, то переносим выполнение текущей операции вперёд на время, равное  $C_{\min}$ , а в текущем цикле переходим к детали с меньшим номером. Трудоёмкость алгоритма составляет  $O(n)$  операций.

## 2. Сложность задачи

В данном разделе исследуется сложность задачи при наличии запретов на простой между некоторыми парами последовательно выполняемых операций.

**Теорема 1.** *Задача составления циклического расписания обработки идентичных деталей с минимальным временем цикла при наличии запретов на простой между некоторыми парами последовательно выполняемых операций NP-трудна в сильном смысле.*

**ДОКАЗАТЕЛЬСТВО.** Полиномиально сведём к рассматриваемой задаче NP-трудную в сильном смысле задачу 3-РАЗБИЕНИЕ [11], которая заключается в следующем. Имеется множество  $3k$  натуральных чисел  $e_1, e_2, \dots, e_{3k}$  и целое число  $E$ , причём  $\frac{E}{4} < e_i < \frac{E}{2}$  и  $\sum_{i=1}^{3k} e_i = kE$ . Можно ли разбить множество этих чисел на  $k$  непересекающихся подмножеств  $T_1, T_2, \dots, T_k$  таких, что  $\sum_{i \in T_j} e_i = E$ ,  $j = 1, \dots, k$ ?

Задачу построения циклического расписания также сформулируем как задачу распознавания: существует ли при заданном  $C_0$  допустимое циклическое расписание с временем цикла, не превосходящим  $C_0$ ? Для сведения к сформулированной задаче 3-РАЗБИЕНИЕ сформируем соответствующую задачу построения циклического расписания. Пусть число машин  $m$  равно 2, а число операций одной детали  $n$  равно  $5k + 1$ . Операции с чётными номерами  $2, 4, 6, \dots, 2k$  выполняются на второй машине, все остальные  $4k + 1$  операций — на первой машине. Длительности первых  $2k + 1$  операций равны  $E$ , а для остальных операций полагаем  $p_{2k+1+i} = e_i$ ,  $i = 1, 2, \dots, 3k$ . Наконец,  $C_0 = (2k + 1)E$ . Пусть  $N_Z = \{1, 2, \dots, 2k\}$ , т. е. первые  $2k + 1$  операций должны выполняться без простоев.

Покажем, что требуемое разбиение существует тогда и только тогда, когда в построенной задаче существует допустимое циклическое расписание.

Пусть такое разбиение  $T_1, T_2, \dots, T_k$  существует. Тогда соответствующее допустимое расписание длины  $C_0$  строится следующим образом. Первые  $2k + 1$  операций детали выполняются в непрерывном режиме в соответствии с требованиями условия задачи. Суммарная длительность

этих операций в точности равна  $(2k + 1)E = C_0$ . Остальные  $3k$  операций, причём предшествующих деталей, выполняются на первой машине в оставшихся  $k$  временных окнах, длина каждого из которых равна  $E$ . Операции по окнам достаточно разместить в соответствии с разбиением  $T_1, T_2, \dots, T_k$  (рис. 1). Если технологический порядок предшествования операций нарушается, то очередная операция берётся из предыдущей детали.

$M_1$	$O_1$	$T_1$	$O_3$	$T_2$	$O_5$	$\dots$	$T_k$	$O_{2k+1}$
$M_2$		$O_2$		$O_4$		$\dots$	$O_{2k}$	

Рис. 1. Схема расписания в промежутке  $[0; C_0]$

Докажем в обратную сторону. Пусть существует допустимое расписание длины  $C_0$ . Так как непрерывно выполняемая последовательность первых  $2k + 1$  операций детали занимает ровно  $C_0$  единиц времени, оставшиеся операции в цикле выполняются параллельно с этой непрерывной цепочкой и заполняют ровно  $k$  окон, каждое длины  $E$ . Множества операций в каждом из этих окон и дают соответствующее 3-разбиение.

Полиномиальность сводимости очевидна. Теорема 1 доказана.

Сложная ситуация с многократным использованием одной машины, описанная в теореме 1, не такой редкий случай на практике. Например, при химической обработке деталей требуется нанести многослойное покрытие, причём слои могут чередоваться и повторяться. Таким образом, выписанный класс примеров задачи моделирует реальную производственную ситуацию.

### 3. Выполнение детали без простоев между операциями

Содержательным является случай задачи, когда обработка любой детали должна происходить без простоев между операциями, т. е. запрет наложен на каждую пару последовательно выполняемых операций. В этом случае  $N_Z = \{1, 2, \dots, n - 1\}$ .

Без ограничения общности можно считать, что первая операция детали выполняется в начале цикла. Это значит, что неизвестным в задаче является лишь время цикла  $C$ . Рассмотрим циклическое расписание со временем цикла  $C$  ( $C \geq C_{\min}$ ). В промежутке времени, равном длине цикла, выполняется весь набор операций, возможно, разных деталей. Так как детали обрабатываются без простоев между операциями, можно сказать, что весь набор операций распределился по  $K$  уровням, где  $K = \lceil P/C \rceil$ ,  $P = \sum_{i=1}^n p_i$ . Первый уровень состоит из операций детали  $K$ ,

второй уровень — из операций детали  $K - 1$ , и т. д., а уровень  $K$  образован операциями первой детали. Каждый уровень, кроме, может быть, последнего, заполнен полностью.

В примере на рис. 2 показано распределение операций детали по уровням, на рис. 3 — соответствующее циклическое расписание, где в первом столбце указан номер соответствующей детали. Одинаковой штриховкой отмечены операции, выполняющиеся на одной машине.

Номер детали	Номер уровня	Распределение по уровням
3	1	$O_1$ $O_2$ $O_3$
2	2	$O_3$ $O_4$ $O_5$ $O_6$
1	3	$O_6$ $O_7$

Рис. 2. Распределение по уровням

1	$O_6$	$O_7$								
2	$O_3$	$O_4$	$O_5$	$O_6$	$O_7$					
3	$O_1$	$O_2$	$O_3$	$O_4$	$O_5$	$O_6$	$O_7$			
4			$O_1$	$O_2$	$O_3$	$O_4$	$O_5$	$O_6$		
5					$O_1$	$O_2$	$O_3$			
	0		$C$			$2C$				

Рис. 3. Циклическое расписание

Если в промежутке  $[0; C]$  не существует такого момента времени, в котором на одной машине выполняется более одной операции (при рассмотрении всех обрабатываемых деталей), то расписание, очевидно, является допустимым. Тогда рассматриваемая задача заключается в следующем: найти наименьшее время цикла  $C^*$ , при котором расписание допустимо в указанном выше смысле.

Обозначим через  $s_j$  время начала операции  $O_j$  некоторой детали в промежутке  $[0; C]$ ,  $j = 1, \dots, n$ . Пусть  $k_j$  — номер уровня, на котором начинается операция  $O_j$ . Величины  $k_j$  и  $s_j$  легко вычисляются:  $k_j = \lfloor \frac{t_j}{C} \rfloor + 1$ ,  $s_j = t_j - (k_j - 1)C$ , где  $t_1 = 0$ ,  $t_j = \sum_{u=1}^{j-1} p_u$ ,  $j = 2, \dots, n$ . Здесь величины  $t_j$  задают времена начала операций для детали  $K$ . Если  $s_j + p_j \leq C$ , то операция  $O_j$  заканчивается на уровне  $k_j$ . В противном случае в промежутке  $[0; C]$  эта операция выполняется только в течение  $C - s_j$  единиц

времени, а остальные  $s_j + p_j - C$  единиц выполняются в начале следующего промежутка  $[C; 2C]$ . При этом в промежутке  $[0; C]$  заканчивается выполнение операции  $O_j$ , но предыдущей детали. Тем самым в промежутке  $[0; s_j + p_j - C]$ , причём на уровне  $k_j + 1$ , заканчивается выполнение операции  $O_j$  предыдущей детали. Можно сказать, что операция  $O_j$  переходит с уровня  $k_j$  на уровень  $k_j + 1$ . В примерах на рис. 2 и 3 такими операциями являются  $O_3$  и  $O_6$ .

Выпишем условия допустимости циклического расписания со временем цикла  $C$  ( $C \geq C_{\min}$ ).

Пусть операции  $O_l$  и  $O_j$  выполняются на одной машине,  $l < j$ . Чтобы не было конфликта, необходимо, чтобы интервалы выполнения этих операций не пересекались.

Если  $s_l > s_j$ , то должны быть верны неравенства

$$s_j + p_j \leq s_l, \quad (1)$$

$$s_l + p_l - C \leq s_j. \quad (2)$$

Выполнение неравенства (1) необходимо для того, чтобы не конфликтовали конец операции  $O_j$  и начало операции  $O_l$ . Выполнение неравенства (2) требуется в случае перехода операции  $O_l$  с одного уровня на следующий; в этом случае не допускается конфликта между концом операции  $O_l$  и началом операции  $O_j$ . Если операция  $O_l$  завершается в том же уровне, в каком и началась, то неравенство (2) выполняется.

Если  $s_l \leq s_j$ , то по аналогии должны быть верны неравенства

$$s_l + p_l \leq s_j, \quad (3)$$

$$s_j + p_j - C \leq s_l. \quad (4)$$

Выполнение неравенства (3) необходимо для того, чтобы не конфликтовали конец операции  $O_l$  и начало операции  $O_j$ . Выполнение неравенства (4) требуется в случае перехода операции  $O_j$  с одного уровня на следующий; в этом случае не допускается конфликта между началом операции  $O_l$  и концом операции  $O_j$ . Если операция  $O_j$  завершается в том же уровне, в каком и началась, то неравенство (4) выполняется.

Для некоторого  $C$  построим циклическое расписание, возможно, недопустимое. Пусть некоторые операции  $O_l$  и  $O_j$  выполняются на одной машине,  $l < j$ . Если их интервалы выполнения пересекаются, будем называть эти операции *конфликтующими*. Чтобы устранить конфликт между такими операциями, необходимо увеличить время цикла на некоторую положительную величину  $\Delta_{lj}$ .

Рассмотрим случай  $s_l > s_j$ . Так как операции конфликтуют, хотя бы одно из неравенств  $s_j + p_j \leq s_l$  и  $s_l + p_l - C \leq s_j$  не выполняется.

Заметим, что оба неравенства неверными быть не могут, так как в этом случае  $C < p_l + p_j$ , что невозможно. Возможны следующие случаи.

1)  $s_l > s_j$ ,  $s_j + p_j > s_l$ . Операцию  $O_j$  необходимо сдвинуть относительно операции  $O_l$  на  $s_j + p_j - s_l$  единиц времени. В этом случае положим  $\Delta_{lj} = \frac{s_j + p_j - s_l}{k_j - k_l}$ . Докажем, что в расписании с длиной цикла  $C + \Delta_{lj}$  операции  $O_l$  и  $O_j$  не конфликтуют. Время начала операции  $O_l$  некоторой детали, расположенной на уровне  $k_l$ , сдвигается влево на время  $\Delta_{lj}(k_l - 1)$ , т. е. время её начала равно  $s_l - \Delta_{lj}(k_l - 1)$ . По аналогии время начала операции  $O_j$  становится равным  $s_j - \Delta_{lj}(k_j - 1)$ . При подстановке этих значений в неравенство (1) оно становится верным равенством, конец операции  $O_j$  совпадает с началом операции  $O_l$ . Заметим, что при сдвиге время начала одной или обеих операций, возможно, выходит за рамки промежутка  $[0; C]$ . Это значит, что операции в промежутке  $[0; C]$  переходят на уровень с меньшим номером. На наличие конфликта между операциями  $O_l$  и  $O_j$  это не влияет, так как в любом цикле конец операции  $O_j$  будет совпадать с началом операции  $O_l$ . Неравенство (2) также будет выполняться.

2)  $s_l > s_j$ ,  $s_l + p_l - C > s_j$ . Это выполняется только в том случае, когда операция  $O_l$  переходит с уровня на уровень и конфликт происходит между концом операции  $O_l$  и началом операции  $O_j$ . В этом случае операцию  $O_j$  необходимо сдвинуть относительно операции  $O_l$  на  $s_j + p_j + C - s_l$  единиц времени. Положим  $\Delta_{lj} = \frac{s_j + p_j - s_l + C}{k_j - k_l - 1}$ . Заметим, что при сдвиге операция  $O_j$  будет выполняться перед операцией  $O_l$  и необходимо выполнение неравенств (3), (4), что проверяется непосредственной подстановкой.

Случай  $s_l \leq s_j$  рассматривается аналогично. В случае нарушения неравенства (3) полагаем  $\Delta_{lj} = \frac{s_j + p_j - s_l}{k_j - k_l}$ , а в случае нарушения неравенства (4) —  $\Delta_{lj} = \frac{s_j + p_j - s_l - C}{k_j - k_l + 1}$ .

Таким образом, во всех случаях после увеличения времени цикла  $C$  на величину  $\Delta_{lj}$  операции  $O_l$  и  $O_j$  перестанут конфликтовать.

Опишем алгоритм построения расписания с минимальным временем цикла при условии выполнения деталей без простоев между операциями.

### Алгоритм А

Вычисляем нижнюю границу времени цикла  $C_{\min}$  и полагаем  $C = C_{\min}$ . Вычисляем  $t_1 = 0$ ,  $t_j = \sum_{u=1}^{j-1} p_u$ ,  $j = 2, \dots, n$ . Полагаем  $w = 1$ .

### Итерация $w$ .

1. Вычисляем  $k_j = \lfloor \frac{t_j}{C} \rfloor + 1$ ,  $s_j = t_j - (k_j - 1)C$ ,  $j = 1, \dots, n$ .



2. Проверяем допустимость расписания.

- Полагаем  $\Delta_{\max} = 0$ ;  $T_i = 0$ ,  $i = 1, \dots, m$ .
- Перебираем операции  $O_j$  в порядке неубывания  $s_j$ .

Если  $s_j < T_{m_j}$ , то операция  $O_j$  конфликтует с предыдущей операцией  $l$ , выполняемой на машине  $m_j$ . В этом случае в зависимости от соотношения величин  $s_l$  и  $s_j$  вычисляем величину  $\Delta_{lj}$  для этих операций по одной из приведённых ранее формул.

Полагаем  $T_{m_j} = s_j + p_j$ .

- Для каждой машины проверяем, конфликтуют ли последняя и первая операции, на ней выполняющиеся.
- Среди всех величин  $\Delta_{lj}$  выбираем максимальную  $\Delta_{\max}$ .

3. Если  $\Delta_{\max} > 0$ , то полагаем  $C = C + \Delta_{\max}$ ;  $w = w + 1$ .

Иначе конец алгоритма.

**Конец итерации  $w$ .**

Оценим трудоёмкость алгоритма. Трудоёмкость каждой итерации составляет  $O(n \log_2 n)$  операций. Одна и та же пара операций может встретиться в конфликтной ситуации несколько раз. Но при повторном конфликте число уровней между операциями сократится. Поэтому не более чем за  $n$  итераций пара попадёт на один уровень и уйдёт из рассмотрения. Так как пар не более чем  $C_n^2$ , число итераций не более чем  $n^3$ . Таким образом, трудоёмкость алгоритма не превышает  $O(n^4 \log_2 n)$ . Тем самым полиномиальная разрешимость данного случая установлена и доказана.

**Теорема 2.** *Задача построения циклического расписания обработки идентичных деталей с минимальным временем цикла при условии выполнения детали без простоев между операциями является полиномиально разрешимой.*

Аналогичный результат следует из работы [12], в которой для задачи составления циклического расписания в роботизированной производственной линии при отсутствии простоев между операциями предложен алгоритм трудоёмкости  $O(n^3 \log_2 n)$ . Данный алгоритм может быть применён для решения нашей задачи.

Для улучшения теоретического результата далее предлагается алгоритм В решения задачи трудоёмкости  $O(n^3)$ . Идея алгоритма состоит в следующем. Для всех пар операций, выполняющихся на одной машине, определим множества (интервалы) тех значений длины цикла, при которых между этими операциями нет конфликта. Наименьшее значение длины цикла из пересечения этих множеств является оптимальным.

Отметим, что идея метода интервалов была предложена ещё в работах [13, 14].

Опишем предлагаемый алгоритм формально. Рассмотрим пару операций  $O_l$  и  $O_j$ , выполняющихся на одной машине (для определённости считаем, что  $l < j$ ). Чтобы между этими операциями не было конфликта, в случае  $s_l > s_j$  должны выполняться неравенства (1), (2), а в случае  $s_l \leq s_j$  — неравенства (3), (4). Приходим к совокупности систем:

$$\left[ \begin{cases} s_l > s_j, \\ s_j + p_j \leq s_l, \\ s_l + p_l - C \leq s_j; \\ s_l \leq s_j, \\ s_l + p_l \leq s_j, \\ s_j + p_j - C \leq s_l. \end{cases} \right.$$

Подставляя  $s_j = t_j - (k_j - 1)C$ ,  $s_l = t_l - (k_l - 1)C$  и решая получившуюся совокупность систем, получаем, что время цикла должно удовлетворять таким неравенствам:

$$\frac{t_j - t_l + p_j}{k_j - k_l + 1} \leq C \leq \frac{t_j - t_l - p_l}{k_j - k_l},$$

$$\frac{t_j - t_l + p_j}{k_j - k_l} \leq C \leq \frac{t_j - t_l - p_l}{k_j - k_l - 1}.$$

Заметим, что номера уровней  $k_j$  и  $k_l$  зависят от времени цикла и при увеличении цикла разность уровней уменьшается. Вообще разность уровней меняется от 1 до  $\Delta k_{\max} = \lfloor \frac{t_j}{C_{\min}} \rfloor - \lfloor \frac{t_l}{C_{\min}} \rfloor$ , где  $\Delta k_{\max} \leq n$ . Если разность уровней равна нулю, то операции очевидным образом не конфликтуют.

С учётом того, что длина цикла не превышает  $P = \sum_{i=1}^n p_i$ , для каждой пары операций  $l$  и  $j$ , выполняющихся на одной машине, существует не более  $n$  допустимых интервалов конечной длины для длины цикла.

Обозначим через  $U$  общее количество пар операций, совместное выполнение которых невозможно. Если  $n_i$  — число операций, выполняющихся на машине  $i$ , то  $U = \sum_{i=1}^m C_{n_i}^2$ . Пусть  $V_u$  — количество допустимых временных интервалов в паре с номером  $u$ . Обозначим интервалы через  $[a_{uv}; b_{uv}]$ , где  $a_{uv}$  — левая, а  $b_{uv}$  — правая границы  $v$ -го по порядку допустимого интервала для пары  $u$ ,  $u = 1, 2, \dots, U$ ,  $v = 1, 2, \dots, V_u$ . Для каждой пары  $u = 1, 2, \dots, U$  допустимые интервалы строятся в соответствии с описанными выше формулами. Формально алгоритм выглядит следующим образом:

```

 $u := 0$ 
for  $i = 1, \dots, m$  do
  for  $l = 1, \dots, n_i - 1$  do
    for  $j = l + 1, \dots, n_i$  do
       $u := u + 1$ 
       $\Delta k_{\max} := \lfloor \frac{t_j}{C_{\min}} \rfloor - \lfloor \frac{t_l}{C_{\min}} \rfloor$ 
       $v := 0$ 
      for  $\Delta k = \Delta k_{\max}, \dots, 1$  do
         $v := v + 1$ ;  $a_{uv} := \frac{t_j - t_l + p_j}{\Delta k + 1}$ ;  $b_{uv} := \frac{t_j - t_l - p_l}{\Delta k}$ 
         $v := v + 1$ ;  $a_{uv} := \frac{t_j - t_l + p_j}{\Delta k}$ 
        if  $\Delta k = 1$  then  $b_{uv} := P$ 
        else  $b_{uv} = \frac{t_j - t_l - p_l}{\Delta k - 1}$ 
        end if
      end for
       $V_u := v$ 
    end for
  end for
end for

```

В результате для каждого  $u = 1, 2, \dots, U$  получим упорядоченную последовательность допустимых интервалов  $[a_{uv}; b_{uv}]$ ,  $v = 1, 2, \dots, V_u$ , т. е.

$$a_{u1} < b_{u1} < a_{u2} < b_{u2} < \dots < a_{uV_u} \leq b_{uV_u} = P.$$

Остаётся найти наименьшее значение  $C$  из пересечения этих множеств. Это можно сделать за  $O\left(n \sum_{i=1}^m C_{n_i}^2\right)$  операций, так как последовательности интервалов упорядочены для каждого  $u$ . Приведём алгоритм.

### Алгоритм В

Для каждого  $u$  через  $l_u$  будем обозначать номер текущего просматриваемого интервала. Первоначально  $l_u = 1$  для всех  $u$ .

1. Находим  $C = \max_{u=1,2,\dots,U} a_{u1}$ .
  2. Пока  $u \leq U$  выполняем цикл:
    - Начиная с  $v = l_u$ , просматриваем интервалы  $[a_{uv}; b_{uv}]$ , пока  $b_{uv}$  не станет бóльшим или равным  $C$ .
    - Полагаем  $l_u$  равным текущему значению  $v$ .
    - Если  $a_{uv} > C$ , то полагаем  $C = a_{uv}$  и  $u = 1$ , иначе  $u = u + 1$ .
- Конец цикла.

При выходе из цикла алгоритм завершает свою работу, текущее значение  $C$  является оптимальным.

Трудоёмкость алгоритма В не превосходит  $O\left(n \sum_{i=1}^m C_{n_i}^2\right)$  операций. Покажем это. Отметим, что для каждой пары конфликтующих операций может быть не более  $n$  интервалов допустимости. Таким образом, трудоёмкость поиска границ интервалов не превосходит  $O\left(n \sum_{i=1}^m C_{n_i}^2\right)$  операций. Тело основного цикла алгоритма В выполняется не более  $U$  раз, где  $U$  — общее количество пар операций, совместное выполнение которых невозможно. Очевидно, что  $U = \sum_{i=1}^m C_{n_i}^2 \leq n^2$ . Трудоёмкость одного цикла для каждого  $i$  не превышает числа интервалов допустимости, т. е. не превышает  $n$ .

Несмотря на большую верхнюю оценку трудоёмкости алгоритм А экспериментально работает лучше алгоритма В. Это можно объяснить тем, что во втором алгоритме оценка трудоёмкости  $O\left(n \sum_{i=1}^m C_{n_i}^2\right)$  достигается.

В первом же алгоритме трудоёмкость одной итерации равна  $O(n \log_2 n)$ , а количество итераций оценивается величиной  $O(n^3)$ . Однако экспериментально число итераций существенно меньше этой величины, и алгоритм А оказывается быстрее.

#### 4. Решение задачи в общем случае

Предложенный выше алгоритм можно обобщить на более широкий класс подзадач. Пусть число пар последовательных операций, между которыми разрешены простои, равно  $W$ . В [2] доказано, что оптимальное циклическое расписание можно искать среди таких расписаний, в которых времена начала операций принимают рациональные значения, при этом знаменатель дроби не превосходит числа операций детали.

Используя этот факт, можно предложить следующий подход к решению рассматриваемой задачи. Если между парой операций не наложен запрет на простой, то этот простой можно рассматривать как самостоятельную операцию с переменной длительностью  $pr_k$ ,  $k = 1, \dots, W$ . При этом  $pr_k \in \left\{\frac{r}{q} \mid q = 1, \dots, n + W; r = 0, \dots, qP\right\}$ , где  $P = \sum_{j=1}^n p_j$ . Для каждого возможного набора длительностей простоев  $pr_k$  достаточно решить задачу предложенным выше полиномиальным алгоритмом. Число всех наборов  $pr_k$  не превосходит  $(P(n + W)^2)^W$ , поэтому общая трудоёмкость алгоритма не превосходит  $O(n^4 \log_2 n (P(n + W)^2)^W)$ . Заметим, что при фиксированном  $W$  алгоритм становится псевдополиномиальным.

**Теорема 3.** *Задача построения циклического расписания обработки идентичных деталей с минимальным временем цикла при наличии запретов на простой между некоторыми парами последовательно выполняемых операций является псевдополиномиально разрешимой, если число пар последовательных операций, между которыми разрешены простои, ограничено фиксированной величиной.*

При произвольном  $W$  алгоритм имеет экспоненциальную трудоёмкость, что согласуется с теоремой 1. Вместе с тем, если запретов нет вообще, задача полиномиально разрешима. Далее проводится исследование задачи при малом числе запретов. Установлено, что уже при двух запретах задача становится NP-трудной.

### 5. Задача с малым числом запретов

Рассмотрим задачу построения циклического расписания с минимальным временем цикла при наличии одного запрета на простой между некоторыми операциями  $O_k$  и  $O_{k+1}$ .

Если эти операции выполняются на одной и той же машине ( $m_k = m_{k+1}$ ), то можно применить алгоритм построения циклического расписания с минимальным временем цикла без ограничений, описанный в разд. 1.

Если операции  $O_k$  и  $O_{k+1}$  с запретом на простой выполняются на разных машинах  $m_k \neq m_{k+1}$ , то минимальное время цикла по-прежнему равно  $C_{\min}$ . В алгоритме сделаем небольшие изменения. Операцию  $O_k$  выполняем в момент начала цикла, в момент её окончания выполняем операцию  $O_{k+1}$ . В свободные промежутки на каждой машине расставляем оставшиеся операции. Если при этом нарушается отношение предшествования операций одной детали, то передвигаем выполнение текущей операции вперёд на время  $C_{\min}$ , а в текущем цикле переходим к детали с меньшим номером. Таким образом, задача с одним запретом полиномиально разрешима за время  $O(n)$ .

При наличии двух запретов задача становится NP-трудной.

**Теорема 4.** *Задача построения циклического расписания с минимальным временем цикла и двумя запретами на простой между операциями NP-трудна.*

**ДОКАЗАТЕЛЬСТВО.** Полиномиально сведём NP-трудную задачу РАЗБИЕНИЕ [11] к рассматриваемой. Задача РАЗБИЕНИЕ заключается в следующем. Имеется множество  $n_0$  целых чисел  $e_1, e_2, \dots, e_{n_0}$  и целое число  $E$ , причём  $\sum_{i=1}^{n_0} e_i = 2E$ . Можно ли разбить множество этих чисел на два непересекающихся подмножества  $T_1, T_2$  таких, что  $\sum_{i \in T_1} e_i = \sum_{i \in T_2} e_i = E$ ?

Задачу построения циклического расписания также сформулируем как задачу распознавания: существует ли при заданном  $C_0$  допустимое циклическое расписание со временем цикла, не превосходящим  $C_0$ ? Сформулируем соответствующую задачу построения циклического расписания. Пусть число машин  $m$  равно 2, а число операций одной детали  $n$  равно  $n_0 + 3$ . Операции  $O_1, O_3, O_4, \dots, O_n$  выполняются на первой машине, а операция  $O_2$  — на второй. Длительности операций таковы:  $p_1 = p_3 = 1$ ,  $p_2 = E$ ,  $p_i = e_i$ ,  $i = 4, \dots, n$ . Полагаем  $C_0 = 2E + 2$ . Пусть  $N_Z = \{1, 2\}$ .

Покажем, что требуемое разбиение существует тогда и только тогда, когда в построенной задаче существует допустимое циклическое расписание со временем цикла, не превосходящим  $C_0$ .

Пусть искомое разбиение существует. Тогда соответствующее допустимое расписание длины  $C_0$  строится следующим образом. Первые три операции детали выполняются в непрерывном режиме в соответствии с требованиями задачи. Суммарная длительность этих операций равна  $E + 2$ . Остальные  $n - 3$  операций, возможно, предшествующих деталей, выполняются на первой машине в два оставшихся свободных временных интервалах, причём длина каждого интервала равна  $E$ . Операции по свободным интервалам достаточно разместить в соответствии с разбиением  $T_1, T_2$ . При этом если технологический порядок предшествования операций нарушается, то очередная операция берётся из предыдущей детали (рис. 4).

$M_1$	$O_1$	$T_1$	$O_3$	$T_2$
$M_2$		$O_2$	$M_1$	

Рис. 4. Схема расписания в промежутке  $[0; C_0]$

Докажем в обратную сторону. Пусть существует допустимое расписание длины, не превосходящей  $C_0$ . Заметим, что длина цикла не может быть меньше  $C_0$ , так как это нижняя граница длины цикла. Таким образом, длина цикла в рассматриваемом расписании равна  $C_0$ . Заметим, что первая машина будет полностью загружена, поэтому расписание на первой машине даёт решение задачи РАЗБИЕНИЕ. В множество  $T_1$  попадут номера операций, выполняемых на первой машине между операциями  $O_1$  и  $O_3$ , а в множество  $T_2$  — номера всех остальных операций. Полиномиальность сводимости очевидна. Теорема 4 доказана.

Если запреты идут не «подряд», то также можно выделить NP-трудный случай:  $N_Z = \{k, l\}$ , где  $k + 1 \neq l$ ,  $m_k = m_l$ ,  $m_{l+1} = m_{k+1}$ ,  $m_k \neq m_{l+1}$ .

Заметим, что все остальные случаи при двух запретах являются полиномиально разрешимыми за  $O(n)$ . Во всех этих случаях операции с запретами можно расположить так, чтобы на каждой машине оставался один промежуток для размещения оставшихся операций.

Открытым остаётся вопрос, при каком количестве запретов задача становится NP-трудной в сильном смысле.

В заключение отметим, что задача построения циклического расписания обработки идентичных деталей с минимальным временем цикла при наличии запретов на простой между некоторыми парами последовательно выполняемых операций труднорешаема. Однако для крайних случаев задачи, когда это ограничение становится преобладающим либо, наоборот, редким, существуют эффективные алгоритмы решения.

Авторы благодарят рецензента за полезные замечания.

#### ЛИТЕРАТУРА

1. Боброва Е. А., Романова А. А., Сервах В. В. Сложность задачи построения циклических расписаний обработки однотипных деталей // Дискрет. анализ и исслед. операций. 2013. Т. 20, № 4. С. 3–14.
2. Романова А. А., Сервах В. В. Оптимизация выпуска однотипных изделий на основе циклических расписаний // Дискрет. анализ и исслед. операций. 2008. Т. 15, № 5. С. 47–60.
3. Hall N. G., Lee T. E., Posner M. E. The complexity of cyclic shop scheduling problems // J. Sched. 2002. Vol. 5. P. 307–327.
4. Hanen C. Study of a NP-hard cyclic scheduling problem: the recurrent job-shop // Eur. J. Oper. Res. 1994. Vol. 71. P. 82–101.
5. McCormick S. T., Rao U. S. Some complexity results in cyclic scheduling // Math. Comput. Modelling. 1994. Vol. 20. P. 107–122.
6. Middendorf M., Timkovsky V. On scheduling cycle shops: classification, complexity and approximation // J. Sched. 2002. Vol. 5. P. 135–169.
7. Roundy R. Cyclic schedules for job shops with identical jobs // Math. Oper. Res. 1992. Vol. 17, No. 4. P. 842–865.
8. Aldakhilallah K. A., Ramesh R. Cyclic scheduling heuristics for a reentrant job shop manufacturing environment // Int. J. Prod. Res. 2001. Vol. 39. P. 2635–2675.
9. Boudoukh T., Penn M., Weiss G. Scheduling job shops with some identical or similar jobs // J. Sched. 2001. Vol. 4. P. 177–199.
10. Brucker P., Kampmeyer T. Tabu search algorithms for cyclic machine scheduling problems // J. Sched. 2005. Vol. 8. P. 303–322.
11. Гэри М., Джонсон Д. Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982.
12. Kats V., Levner E. Cyclic scheduling in a robotic production line // J. Sched. 2002. Vol. 5. P. 23–41.

- 
13. Айзенштат В. С. Многооператорные циклические процессы // Докл. АН БССР. 1963. Т. VII, № 4. С. 224–227.
14. Танаев В. С. К задаче составления расписания работы поточной линии с одним автооператором // Инж.-физ. журн. 1964. Т. 7, № 3. С. 111–114.

Романова Анна Анатольевна  
Сервах Владимир Вицентьевич

Статья поступила  
27 августа 2018 г.  
После доработки —  
29 июля 2019 г.  
Принята к публикации  
28 августа 2019 г.



COMPLEXITY OF CYCLIC JOB SHOP SCHEDULING  
PROBLEMS FOR IDENTICAL JOBS WITH NO-WAIT  
CONSTRAINTS<sup>\*)</sup>A. A. Romanova<sup>1,a</sup> and V. V. Servakh<sup>2,b</sup><sup>1</sup> Omsk State University,  
55a Mir Avenue, 644077 Omsk, Russia<sup>2</sup> Omsk Branch of Sobolev Institute of Mathematics,  
13 Pevtsov Street, 644099 Omsk, RussiaE-mail: <sup>a</sup>anna.a.r@bk.ru, <sup>b</sup>svv\_usa@rambler.ru

**Abstract.** We consider the cyclic job shop problem with no-wait constraints which consists in minimizing the cycle time. We assume that a single product is produced on a few machines. A job is processed by performing a given set of operations in a predetermined sequence. Each operation can be performed on exactly one machine. We consider the problem of minimization the cycle time with no-wait constraints between some pairs of sequential operations and investigate the complexity of the problem and some of its subproblems. In general, the problem is proved to be strongly NP-hard. In the case when the job is processed without downtime between operations, polynomial solvability is proved and the two algorithms are proposed. Also we develop an algorithm for the general case which is pseudopolynomial if the number of admissible downtime is fixed. The case of a single no-wait constraint is polynomially solvable. The problem with two no-wait constraints becomes NP-hard. We found effectively solvable cases and propose the corresponding algorithms. Illustr. 4, bibliogr. 14.

**Keywords:** scheduling theory, cyclic job shop, identical jobs, computational complexity theory, polynomial algorithm, pseudopolynomial algorithm.

---

<sup>\*)</sup> The research by V. V. Servakh is supported by the Programme for Fundamental Scientific Research of SB RAS No. I.5.1 (project 0314–2019–0019).

## REFERENCES

1. **E. A. Bobrova, A. A. Romanova, V. V. Servakh**, The complexity of the cyclic job shop problem with identical jobs, *Diskretn. Anal. Issled. Oper.* **20** (4), 3–14 (2013) [Russian].
2. **A. A. Romanova and V. V. Servakh**, Optimization of identical jobs production on the base of cyclic schedules, *Diskretn. Anal. Issled. Oper.* **15** (4), 47–60 (2008) [Russian].
3. **N. G. Hall, T. E. Lee, and M. E. Posner**, The complexity of cyclic shop scheduling problems, *J. Sched.* **5** (4), 307–327 (2002).
4. **C. Hanen**, Study of an NP-hard cyclic scheduling problem: The recurrent job-shop, *Eur. J. Oper. Res.* **71**, 82–101 (1994).
5. **S. T. McCormick and U. S. Rao**, Some complexity results in cyclic scheduling, *Math. Comput. Modelling* **20**, 107–122 (1994).
6. **M. Middendorf and V. Timkovsky**, On scheduling cycle shops: Classification, complexity, and approximation, *J. Sched.* **5** (2), 135–169 (2002).
7. **R. Roundy**, Cyclic schedules for job shops with identical jobs, *Math. Oper. Res.* **17** (4), 842–865 (1995).
8. **K. A. Aldakhilallah and R. Ramesh**, Cyclic scheduling heuristics for a reentrant job shop manufacturing environment, *Int. J. Prod. Res.* **39**, 2635–2675 (2001).
9. **T. Boudoukh, M. Penn, and G. Weiss**, Scheduling job shops with some identical or similar jobs, *J. Sched.* **4**, 177–199 (2001).
10. **P. Brucker and T. Kampmeyer**, Tabu search algorithms for cyclic machine scheduling problems, *J. Sched.* **8**, 303–322 (2005).
11. **M. R. Garey and D. S. Johnson**, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (Freeman, San Francisco, 1979; Mir, Moscow, 1982 [Russian]).
12. **V. Kats and E. Levner**, Cyclic scheduling in a robotic production line, *J. Sched.* **5** (1), 23–41 (2002).
13. **V. S. Aizenshtat**, Multi-operator cyclic processes, *Dokl. Nats. Akad. Nauk Belarus* **7** (4), 224–227 (1963) [Russian].
14. **V. S. Tanaev**, A scheduling problem for a flowshop line with a single operator, *Inzh.-Fiz. Zh.* **7** (3), 111–114 (1964) [Russian].

Anna A. Romanova  
Vladimir V. Servakh

Received August 27, 2018  
Revised July 29, 2019  
Accepted August 28, 2019