

ПОСТРОЕНИЕ ЦИКЛИЧЕСКИХ РАСПИСАНИЙ ПРИ НАЛИЧИИ ПАРАЛЛЕЛЬНЫХ МАШИН *)

Е. А. Боброва^а, В. В. Сервах^б

Институт математики им С. Л. Соболева СО РАН, Омский филиал,
ул. Певцова, 13, 644043 Омск, Россия

E-mail: ^аeabobrova88@gmail.com, ^бsvv_usa@rambler.ru

Аннотация. Рассматривается задача обработки партии идентичных деталей со сложным технологическим маршрутом при наличии параллельных машин. Требуется построить циклическое расписание с минимальной длиной цикла при ограничении на максимальное число одновременно обрабатываемых деталей. Предложен и обоснован алгоритм построения точного решения, выделен псевдополиномиально разрешимый случай задачи. Ил. 4, библиогр. 16.

Ключевые слова: циклическое расписание, динамическое программирование, псевдополиномиальный алгоритм.

1. Постановка задачи

На производственной линии необходимо обработать партию из N идентичных деталей. Все детали проходят одинаковый технологический маршрут обработки, состоящий из n последовательно выполняемых операций O_1, O_2, \dots, O_n . Задано множество типов машин $\{1, 2, \dots, M\}$ и μ_m — число машин типа m . Операция O_i выполняется на машине типа m_i непрерывно в течение $p_i \in \mathbb{N}$ единиц времени, $i = 1, 2, \dots, n$, $m_i \in \{1, 2, \dots, M\}$. Одновременное выполнение двух и более операций на одной машине не допускается. Если множество типов машин и множество операций сопоставляются взаимно однозначно, то имеем классический конвейер. Однако в современной экономике быстро меняющийся ассортимент выпускаемой продукции требует оперативной и своевременной переналадки производственной линии, что привело к внедрению дорогостоящих универсальных рабочих станций, способных выполнять множество различных операций. На такие машины деталь при обработке

*) Работа выполнена при финансовой поддержке проекта РАН «Современные методы аппроксимиремости моделей, алгоритмов и теорий» (0314–2015–0009).

может поступать неоднократно. Похожая ситуация возникает при сборке самолётов и ракетоносителей, когда одна бригада многократно возвращается на объект для проведения очередного монтажа или контроля по своему профилю.

Обозначим через t_{ij} время начала выполнения операции O_i детали j . Совокупность $\{t_{ij} \mid i = 1, 2, \dots, n, j = 1, 2, \dots, N\}$ называется *расписанием*. Расписание *допустимо*, если соблюдается технологический порядок, операции выполняются непрерывно без пересечений на машинах. Расписание называется *циклическим*, если для любых $i = 1, 2, \dots, n$ и $j = 1, 2, \dots, N$ выполняется $t_{ij} = t_i + C(j - 1)$, где $t_i = t_{i,1}$ и C — *длина цикла (циклическое время)*. Циклическое расписание определяется заданием длины цикла C и времени t_i начала выполнения каждой операции $i = 1, 2, \dots, n$ для первой детали. Циклические расписания играют важную роль в производстве, обеспечивают ритмичность использования оборудования и исполнителей, способствуют удобному планированию поставок ресурсов и сбыта продукции. Исследованию циклических расписаний посвящено множество работ [4, 8–14, 16].

Задача построения расписания с наименьшей длиной цикла полиномиально разрешима [7]. В пределе, при большом количестве деталей, минимальное циклическое время обеспечивает наибольшую производительность линии. Асимптотически точные алгоритмы минимизации общего времени обработки однотипных деталей на основе циклических расписаний описаны в [5, 6]. Но имеются и негативные моменты использования расписаний с минимальным значением C . Максимальная загрузка машин приводит к тому, что в обработке могут находиться одновременно много деталей и, как следствие, возникают простои между операциями. Это требует транспортировки и хранения незавершённых деталей, большей потребности в оборотных ресурсах, что в конечном итоге приводит к дополнительным затратам.

Иногда число деталей, одновременно находящихся в обработке, ограничено технологически. Например, количество площадок при сборке самолётов фиксировано, и, пока площадка не освободится, невозможно начать сборку очередного самолёта. В автомобильной промышленности при химической обработке на деталь наносится многослойное покрытие, для чего она крепится на оснастке и последовательно опускается в соответствующие резервуары. Количество одновременно обрабатываемых деталей ограничено числом оснасток. Аналогичные постановки возникают в швейной и других отраслях промышленности.

Когда имеется ровно одна машина каждого типа, т. е. $\mu_m = 1$, $m \in$

$\{1, 2, \dots, M\}$, и максимальное число деталей, одновременно находящихся в обработке, не превосходит H , задача минимизации длины цикла является NP-трудной в сильном смысле [9]. Для случая фиксированного H были предложены псевдополиномиальный алгоритм и вполне полиномиальная аппроксимационная схема [2]. В [1] при условии $H = 2$ построен алгоритм полиномиальной трудоёмкости, доказана NP-трудность задачи при $H > 3$. Сложность задачи при $H = 3$ остаётся неизвестной.

На практике возникают ситуации, когда загрузка некоторой машины, т. е. суммарная длительность выполняемых на этой машине операций, существенно больше загрузки остальных. При химической обработке деталей в автомобильной промышленности большая часть операций технологического процесса занимает 3–5 минут, тогда как на операцию хромирования требуется около 18 минут. Если установить дополнительно 2 резервуара для хромирования, то при наличии достаточного числа оснасток можно уменьшить длину цикла в три раза.

Рассмотрим иллюстративный пример. Пусть обработка каждой детали состоит из трёх операций, длительности которых равны 7, 12 и 3 единиц времени, и они выполняются на машинах m_1, m_2 и m_3 соответственно. В случае $\mu_i = 1, i = 1, 2, 3$, оптимальное расписание (с длиной цикла, равной 12) будет выглядеть, как на рис. 1. При добавлении параллельных

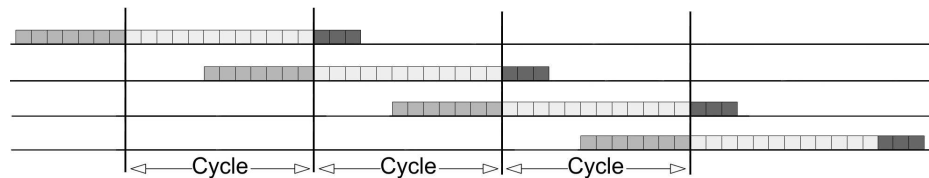


Рис. 1. Расписание без параллельных машин

ельной машины типа m_2 наиболее загруженной становится машина m_1 и оптимальная длина цикла будет равна 7. Соответствующее расписание представлено на рис. 2.

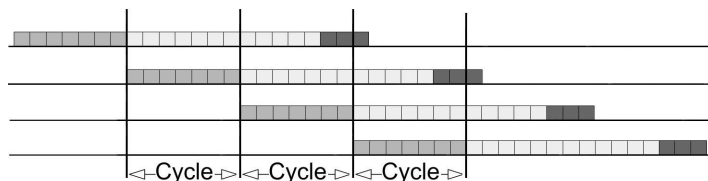


Рис. 2. Расписание с использованием параллельных машин

В данной статье рассматривается задача минимизации циклического времени обработки партии идентичных деталей со сложным технологическим маршрутом, заданным числом машин μ_i каждого типа и ограничении на максимальное число деталей H , одновременно находящихся в обработке. В разд. 2 описывается сведение задачи к последовательности разномаршрутных задач теории расписаний и обосновывается наличие оптимального расписания с определёнными свойствами. В разд. 3 представлен алгоритм решения задачи, основанный на схеме динамического программирования. Выделен псевдополиномиально разрешимый случай.

2. Свойства задачи и обоснование алгоритма её решения

Опишем сведение задачи минимизации длины цикла к решению серии разномаршрутных задач теории расписаний, которое будет использовано в алгоритме, где каждая из разномаршрутных задач решается методом динамического программирования.

Если циклическое расписание разрезать на повторяющиеся циклы, то границы циклов разрежут цепочку операций O_1, O_2, \dots, O_n каждой работы на $d \leq H$ фрагментов, лежащих в d последовательных циклах. Внутри каждого цикла все d фрагментов представлены ровно по одному разу, причём разные фрагменты принадлежат разным деталям. На рис. 3 изображено циклическое расписание при $d = 2$. Каждый фрагмент

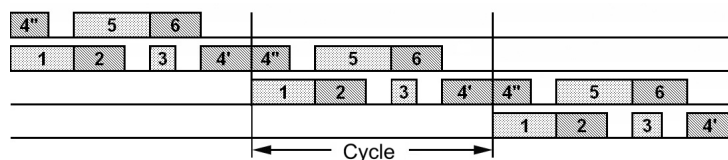


Рис. 3. Циклическое расписание при $H = 2$

может быть интерпретирован как отдельное задание в разномаршрутной задаче теории расписаний. Если минимизировать общее время выполнения этих d заданий (makespan), то и длина цикла при данном разбиении будет минимальной. Далее будем считать, что $H \geq 2$, так как случай $H = 1$ тривиален.

Под разбиением будем понимать вектор $R_d = (\xi_0, \xi_1, \dots, \xi_d)$, удовлетворяющий условию $0 = \xi_0 < \xi_1 < \dots < \xi_d = \sum_{i=1}^n p_i$. Значения ξ_j будем называть *точками разбиения*. Через $T_i = \sum_{l=1}^i p_l$, $i = 1, 2, \dots, n$,

обозначим суммарную длительность первых i операций одной детали, $T = \{0, T_1, T_2, \dots, T_n\}$. Если $\xi_j \in T$, то точка разбиения лежит между операциями. Если $\xi_j \notin T$, то некоторая операция разрезается и её полученные части в силу непрерывности выполнения операций должны примыкать к границам цикла.

Множество индексов операций, содержащих внутренние точки разбиения, обозначим через I^{div} . Каждая операция $O_i \in I^{\text{div}}$ разделена на две или более частей: начальный фрагмент операции O_i^{begin} , конечный фрагмент операции O_i^{end} и $r_i \leq \min\{\mu_i - 1, H - 2\}$ фрагментов O_i^{middle} между ними, каждый из которых занимает весь цикл и имеет длину C . Следовательно, длина операции p_i равна сумме длительности O_i^{begin} , длительности O_i^{end} и $r_i * C$, $i \in I^{\text{div}}$. Фрагмент операции O_i^{begin} должен завершиться в момент окончания цикла, а фрагмент операции O_i^{end} должен начать выполнение в момент начала цикла. На рис. 4 показана схема выполнения одной операции в цикле при $r_i = 3$. Конкретное расписание зависит от длительности O_i^{begin} и длины цикла C .

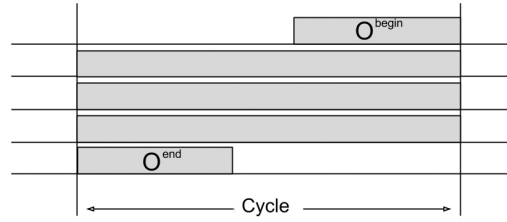


Рис. 4. Разбиение операции на части

Через $I^{\text{whole}} = I \setminus I^{\text{div}}$ обозначим множество индексов операций без точек разбиения. Такие операции внутри цикла могут размещаться свободно, с учётом последовательности операций одного фрагмента.

Полученная для фиксированного разбиения разномаршрутная задача может быть решена алгоритмом динамического программирования, описанным в разд. 3. Остаётся выбрать разбиение, для которого длина цикла минимальна. Однако разбиений бесконечно много. Ниже будет показано, что для поиска оптимума исходной задачи достаточно перебрать конечное множество разбиений. Суть доказательства следующая. Пусть дано некоторое оптимальное решение. Зафиксируем для него порядок выполнения операций в виде графа предшествования $G = (V, E)$, где множество вершин $V = \{O_i^{\text{begin}} \cup O_i^{\text{end}}, i \in I^{\text{div}}\} \cup \{O_i, i \in I^{\text{whole}}\}$, а множество рёбер E задаёт порядок выполнения операций в цепочках и на машинах. Рассматриваются только графы, в которых операции O_i^{end}

стоят первыми в своих цепочках и на своих машинах, а операции из O_i^{begin} стоят последними в своих цепочках и на своих машинах, $i \in I^{\text{div}}$. Если две операции выполняются на машине одного типа и времена их выполнения не пересекаются, то в E вводим соответствующую дугу. Далее, длительности операций O_i^{begin} , длину цикла и времена начала операций из I^{whole} делаем переменными. Требуется найти расписание, в котором время завершения выполнения всех операций минимально. Для фиксированного порядка выполнения работ на машинах получаем задачу линейного программирования. Ниже показано, что среди множества оптимальных решений такой задачи найдётся решение с определёнными свойствами, а разбиений с такими свойствами конечное число. Будем перебирать только их. Заметим, что модель линейного программирования используется при обосновании алгоритма, а не при решении задачи.

Опишем переменные задачи. Пусть s_i — время начала выполнения операции O_i , $i \in I^{\text{whole}}$, или операции O_i^{begin} , $i \in I^{\text{div}}$, x_i — длительность операции O_i^{begin} , удовлетворяющая условию $x_i \leq p_i - r_i * C_{\text{max}}$, $i \in I^{\text{div}}$, а C_{max} — общее время завершения выполнения всех операций.

Операция O_i начинается в момент времени s_i и имеет длительность p_i , $i \in I^{\text{whole}}$. Операции O_i^{begin} , $i \in I^{\text{div}}$, начинаются в момент времени s_i , имеют длительность x_i и заканчиваются в момент времени C_{max} . Операции O_i^{end} , $i \in I^{\text{div}}$, начинаются в момент времени 0 и имеют длительность $p_i - x_i - r_i * C_{\text{max}}$. Средние части операций O_i , $i \in I^{\text{div}}$, начинаются в момент времени 0 и завершаются в момент времени C_{max} .

Разобьём множество рёбер E на следующие подмножества:

$$\begin{aligned} E_1 &= \{(O_i, O_j) \in E \mid i \in I^{\text{whole}}, j \in I^{\text{whole}}\}, \\ E_2 &= \{(O_i, O_j^{\text{begin}}) \in E \mid i \in I^{\text{whole}}, j \in I^{\text{div}}\}, \\ E_3 &= \{(O_i^{\text{end}}, O_j) \in E \mid i \in I^{\text{div}}, j \in I^{\text{whole}}\}, \\ E_4 &= \{(O_i^{\text{end}}, O_j^{\text{begin}}) \in E \mid i \in I^{\text{div}}, j \in I^{\text{div}}\}. \end{aligned}$$

Рёбрам из E соответствуют следующие условия предшествования:

$$s_i + p_i \leq s_j, \quad (O_i, O_j) \in E_1, \quad (O_i, O_j^{\text{begin}}) \in E_2,$$

$$p_i - x_i - r_i * C_{\text{max}} \leq s_j, \quad (O_i^{\text{end}}, O_j) \in E_3, \quad (O_i^{\text{end}}, O_j^{\text{begin}}) \in E_4.$$

Последние операции в цепочках должны завершиться не позднее момента времени C_{max} :

$$s_i + p_i \leq C_{\text{max}}, \quad i \in I^{\text{whole}}, \quad s_i + x_i = C_{\text{max}}, \quad i \in I^{\text{div}}.$$

Модель линейного программирования для разномаршрутной задачи теории расписаний с критерием минимизации общего времени выполнения требования и дополнительными условиями на непрерывное выполнение операций исходной задачи выглядит следующим образом:

$$C_{\max} \rightarrow \min \quad (1)$$

$$C_{\max} - s_i \geq p_i, \quad i \in I^{\text{whole}}, \quad (2)$$

$$C_{\max} - s_i - x_i = 0, \quad i \in I^{\text{div}}, \quad (3)$$

$$s_j - s_i \geq p_i, \quad (O_i, O_j) \in E_1, \quad (O_i, O_j^{\text{begin}}) \in E_2, \quad (4)$$

$$r_i * C_{\max} + s_j + x_i \geq p_i, \quad (O_i^{\text{end}}, O_j) \in E_3, \quad (O_i^{\text{end}}, O_j^{\text{begin}}) \in E_4, \quad (5)$$

$$-r_i * C_{\max} - x_i \geq -p_i, \quad i \in I^{\text{div}}, \quad (6)$$

$$s_i \geq 0, \quad i \in I^{\text{whole}} \cup I^{\text{div}}, \quad (7)$$

$$x_i \geq 0, \quad i \in I^{\text{div}}. \quad (8)$$

Условие (1) соответствует критерию оптимизации. В (2) записано то, что операции, не содержащие точек разбиения, должны завершиться не позже конца цикла. Условие (3) говорит о том, что выполнение операции O_i^{begin} , $i \in I^{\text{div}}$, заканчивается в момент конца цикла. В (4) и (5) записаны условия предшествования операций, а (6) гарантирует, что операции из I^{div} имеют достаточную длину, чтобы содержать r_i срединных частей.

Лемма. Пусть A — матрица ограничений задачи (1)–(8). Любой минор матрицы A не превосходит по модулю $H^2 + H$.

Доказательство. Обозначим через M произвольный ненулевой минор матрицы A , а через L — его порядок. Покажем, что абсолютное значение M не превосходит $H^2 + H$. Для того чтобы оценить минор M , рассмотрим его разложение по последнему столбцу: $M = \sum_{i=1}^L (-1)^{i+L} a_{iL} M_{iL}$. Здесь M_{iL} — определитель матрицы, полученной из матрицы A удалением строки i и столбца L .

Разобьём получившуюся матрицу на блоки. Разделим столбцы на два типа: S_{i_v} , соответствующие переменным s_{i_v} , $v = 1, \dots, L_1$, и X_{j_u} , соответствующие x_{j_u} , $u = 1, \dots, L - L_1 - 1$. Сгруппируем строки в порядке (1)–(8). Тогда разбиение на блоки примет следующий вид:

| S_{i_v} | X_{j_u} | C_{\max} |
|-----------|-----------|------------|
| B_1 | 0 | 1 |
| B_2 | B_3 | 1 |
| B_4 | 0 | 0 |
| B_5 | B_6 | r_j |
| 0 | B_7 | $-r_j$ |

В данном представлении 0 соответствует блоку, состоящему из нулей. Строки блока B_1 соответствуют ограничению (2), и значения -1 встречаются только для переменной s_j . Таким образом, в каждой строке блока B_1 содержится не более одного значения -1 . В каждой строке блоков B_2 и B_3 содержится не более одного значения -1 , так как они соответствуют ограничениям типа (3). Блок B_4 представляет собой ограничения типа (4), и в нём коэффициент 1 при переменных s_j , а при переменных s_i — коэффициент -1 . Каждая строка блока B_4 содержит не больше одной 1 и не больше одной -1 . Каждая строка блоков B_5 и B_6 содержит не более одного значения, равного 1, так как в ограничениях типа (5) переменные s_j и x_i имеют коэффициент 1. И наконец, ограничения последнего типа (6) соответствуют блоку из нулей и блоку B_7 . В каждой строке блока B_7 содержится не более одного значения -1 . В каждой строке матрицы минора M_{iL} содержится не более двух ненулевых элементов. Для того чтобы оценить минор M_{iL} , разложим его по строкам и столбцам с единственными ненулевыми значениями 1 или -1 . Получим одно из чисел 1, 0, -1 или некоторую матрицу. Рассмотрим случай, когда размер оставшейся матрицы равен $l \geq 1$. Эта матрица может содержать только строки из блоков B_2 , B_3 , B_4 , B_5 и B_6 . Если в матрице нет столбцов из B_3 или B_6 , то раскладываем определитель матрицы по строкам из B_2 и B_5 . Тогда в матрице остаются только строки и столбцы блока B_4 . Напомним, что каждая строка этой матрицы содержит не более одного значения 1 и одного значения -1 . Следовательно, сумма всех столбцов такой матрицы содержит нулевой столбец. Если матрица содержит столбцы из блоков B_3 или B_6 , то определитель становится равен нулю. Действительно, рассмотрим следующую линейную комбинацию столбцов: $\sum_{v=1}^{l_1} S_{i_v} - \sum_{u=1}^{u-l_1} X_{j_u}$. Исходя из структуры матрицы, получаем нулевой столбец. Следовательно, столбцы матрицы линейно зависимы и каждый минор матрицы M_{iL} равен 0, -1 или 1. Последний столбец матрицы A содержит не более H единиц, так как переменная C_{\max} встречается только в ограничениях, связанных с последними операциями в цепочках, а число цепочек не превосходит H . Последний стол-

бец матрицы A также содержит r_j не более H раз по тем же причинам, и справедлива оценка

$$|M| = \left| \sum_{i=1}^L (-1)^{i+L} a_{iL} M_{iL} \right| \leq (1 + \max(r_j)) * H \leq H^2 + H.$$

Лемма доказана.

Теорема 1. Для задачи минимизации длины цикла при условии, что число одновременно обрабатываемых деталей не превосходит H , и наличии параллельных машин существуют оптимальное расписание $\{t_i\}$, $i = 1, 2, \dots, n$, и целое число $q \leq H^2 + H$ такие, что все t_i и минимальная длина цикла C^* кратны $\frac{1}{q}$.

ДОКАЗАТЕЛЬСТВО. Оптимальное решение задачи (1)–(8) существует и соответствует вершине многогранника допустимых решений. Следовательно, существует решение системы $Bu = b$, где B — матрица, соответствующая ненулевому минору целочисленной матрицы A , b — целочисленный вектор и u — вектор базисных переменных. Отсюда знаменатель в $u = B^{-1}b$ не может быть больше чем $H^2 + H$, так как по лемме $|B| \leq H^2 + H$. Более того, поскольку все компоненты u получены делением на некоторый минор $|B|$, все они кратны $\frac{1}{|B|}$, включая $C^* = C_{\max}$. Моменты начала выполнения операций t_i , $i = 1, 2, \dots, n$, и длина цикла C^* могут быть тривиальным образом получены из значений вектора u и тоже кратны $\frac{1}{|B|}$. Теорема 1 доказана.

Теорема 1 даёт возможность использовать динамическое программирование для решения рассматриваемой задачи.

3. Алгоритм динамического программирования

Алгоритм заключается в переборе всех разбиений, построенных в соответствии с результатом предыдущего раздела и решении для каждого разбиения разномаршрутной задачи. Перебираем $q = 1, 2, \dots, H^2 + H$. Умножим длительности операций на q : $p_i := p_i * q$, $i = 1, 2, \dots, n$, и далее будем рассматривать только целочисленные разбиения. Отметим, что если в разбиении существуют серединные операции O_i^{middle} , то все они должны иметь одинаковую длительность, равную длине цикла. В этом случае требуется построить допустимое расписание фиксированной длины. При разбиении R_d имеется $d \leq H$ цепочек операций. Последовательность операций одной цепи будем называть заданием. Обозначим через S_k сумму длительностей операций задания $k = 1, 2, \dots, d$. Требуется

найти расписание с минимальной длиной цикла, удовлетворяющее ограничениям на непрерывность выполнения операций и на одновременное выполнение не более чем μ_m операций на машинах типа m . Для решения этой разномаршрутной задачи теории расписаний в [3,15] был предложен алгоритм динамического программирования. Основная идея алгоритма заключается в переборе промежуточных состояний выполнения заданий и поиске для каждого состояния лучшего частичного расписания.

Обозначим через x_k текущее состояние выполнения операций задания k . Если $x_k = 0$, то выполнение операций этого задания ещё не началось. Если $x_k = S_k$, то задание k завершилось, $k = 1, 2, \dots, d$. Промежуточное значение x_k означает, что общее время выполнения операций этого задания без учёта простоев составило x_k и требуется ещё $S_k - x_k$ единиц времени для его завершения. Вектор $\mathbf{x} = (x_1, x_2, \dots, x_d)$ называется *состоянием* выполнения заданий. Вектор $\mathbf{0} = (0, 0, \dots, 0)$ — начальное состояние, при котором ни одна операция не начала своего выполнения, $\mathbf{S} = (S_1, S_2, \dots, S_d)$ — конечное состояние, т. е. все задания выполнены. Через $X = \{\mathbf{x} = (x_1, x_2, \dots, x_d) \mid x_k \in \{0, 1, \dots, S_k\}, k = 1, 2, \dots, d\}$ обозначим множество всех состояний.

Переход из одного состояния в другое осуществляется с помощью соответствующего управления. *Управление* — это булев вектор $\delta = (\delta_1, \delta_2, \dots, \delta_d)$, где $\delta_k \in \{0, 1\}$, $k = 1, 2, \dots, d$. При этом из состояния \mathbf{x} переходим в состояние $\mathbf{x} + \delta$. Процесс перехода соответствует одновременному выполнению в единичном временном интервале операций тех заданий, для которых $\delta_k = 1$. Если состояние \mathbf{x} соответствует тому, что операция некоторого задания k была начата и ещё не завершена, то в силу непрерывности выполнения операций δ_k должно быть равно 1. При этом переход $\mathbf{x} \rightarrow \mathbf{x} + \delta$ для управления δ , у которого $\delta_k = 0$, будет недопустим. Управление δ *недопустимо*, если количество операций, для которых $\delta_k = 1$, выполняемых на одном типе машин, будет больше числа машин этого типа. Дополнительные ограничения, связанные с необходимостью непрерывного выполнения тех операций, в которых производится разрез, учитываются следующим образом. Все допустимые управления δ из состояния $\mathbf{0}$ должны удовлетворять соотношению $\delta_k = 1$, если задание k начинается с выполнения операции O_i^{end} , $i \in I^{\text{div}}$. Аналогично для всех допустимых управлений δ , переводящих систему в состояние \mathbf{S} , должно быть выполнено $\delta_k = 1$, если задание k заканчивается выполнением операции O_i^{begin} , $i \in I^{\text{div}}$. Необходимо найти последовательность допустимых управлений, которая за минимальное число переходов позволяет перейти из состояния $\mathbf{0}$ в состояние \mathbf{S} . Множество допустимых управлений, при-

водящих в состояние \mathbf{x} , обозначим через $\Delta_{\mathbf{x}}$. Пусть $L(\mathbf{x})$ — наименьшее число переходов из состояния $\mathbf{0}$ в состояние \mathbf{x} . Выпишем рекуррентное соотношение:

$$L(\mathbf{0}) = 0, \quad L(\mathbf{x}) = \min_{\delta \in \Delta_{\mathbf{x}}} \{L(\mathbf{x} - \delta) + 1\}, \quad \mathbf{x} \in X \setminus \{\mathbf{0}\}.$$

Предлагаемый алгоритм реализует стандартную схему динамического программирования. Алгоритм начинает свою работу с начального состояния $\mathbf{0}$ и в порядке лексикографического возрастания перебирает все состояния $\mathbf{x} \in X$, вычисляя значения $L(\mathbf{x})$ по выписанной формуле. При этом запоминаем вектор $\delta(\mathbf{x})$, на котором получено оптимальное значение $L(\mathbf{x})$. Величина $L(\mathbf{S})$ определяет оптимальное значение целевой функции. Восстановление оптимального решения осуществляется стандартным способом.

Приведём схему алгоритма, решающего задачу для фиксированного разбиения.

```

procedure  $DP(d, S_1, S_2, \dots, S_d, REZ)$ ;
do  $k = 1$  to  $d$   $x_k = 0$ ;  $L(\mathbf{0}) = 0$ ;
do while  $k \neq 0$  begin
     $x_d = x_d + 1$ ;  $k = d$ ;
    do while  $x_k = S_k + 1$  begin
         $x_k = 0$ ;  $k = k - 1$ ;  $x_k = x_k + 1$ ; end;
    do  $i = 1$  to  $d$   $\delta_i = 0$ ;
    do while  $i \neq 0$  begin
         $\delta_d = \delta_d + 1$ ;  $i = d$ ;
        do while  $\delta_i = 2$  begin  $\delta_i = 0$ ;  $i = i - 1$ ;  $\delta_i = \delta_i + 1$ ; end;
        if {состояние  $\mathbf{x}$  и переход  $\delta$  удовлетворяют ограничению
            непрерывности выполнения операций}
            & {вектор  $\delta$  удовлетворяет ограничению на число машин
                каждого типа}
        then  $L(\mathbf{x}) = \min\{L(\mathbf{x}), L(\mathbf{x} - \delta) + 1\}$ ;
    end {do while  $i \neq 0$ };
end {do while  $k \neq 0$ };
 $REZ = L(S_1, S_2, \dots, S_k)$ ;
end{procedure}.

```

Для того чтобы проверить, что переход δ допустим для состояния \mathbf{x} , нужно рассмотреть несколько условий. Из непрерывности выполнения

операций следует, что операция O_i^{end} начинает выполняться в момент начала цикла, окончание операции O_i^{begin} совпадает с концом цикла и срединные операции O_i^{middle} выполняются в течении всего цикла. Следовательно, если $\mathbf{x} - \delta = \mathbf{0}$ и задание k начинается с операции O_i^{end} или O_i^{middle} , то $\delta_k = 1$. В противном случае переход $\mathbf{0} \rightarrow \delta$ недопустим. Аналогично, когда задание k заканчивается операцией O_i^{begin} или O_i^{middle} , то переход $\mathbf{S} - \delta \rightarrow \mathbf{S}$ недопустим, если $\delta_k = 0$. Для любого перехода $\mathbf{x} - \delta \rightarrow \mathbf{x}$ в состоянии $\mathbf{x} - \delta$, если операция i задания k начала выполнение, но ещё не закончена, то $\delta_k = 1$. Проверка того, что выполняется ограничение на количество машин каждого типа, тривиальна.

Оценим время работы алгоритма. Имеется три уровня вычислений. На первом уровне фиксируем q . Количество возможных значений q равно $H^2 + H$. На втором уровне для фиксированного q рассматриваются все возможные C_{Pq}^{H-1} разбиений, где $P = \sum_{i=1}^n p_i$ — суммарная длительность всех операций одной детали. На третьем уровне решается разномаршрутная задача теории расписаний при фиксированных q и разбиении. Для каждого состояния $\mathbf{x} \in X$ необходимо вычислить $L(\mathbf{x})$, что требует перебора не более $2^H - 1$ булевых векторов δ . При этом проверка допустимости вектора требует порядка $O(H^2)$ операций. Всего получаем не более $O(2^H H^2 S_1 S_2 \dots S_H)$ операций. Заметим, что $S_1 S_2 \dots S_H \leq (qP/H)^H$ и $qP/H \leq P$. Следовательно, трудоёмкость алгоритма динамического программирования для одного разбиения не превосходит $O(2^H H^2 P^H)$ операций.

Число возможных разбиений при фиксированном q не превышает C_{Pq}^{H-1} . По формуле Стирлинга с учётом $q \leq H$ получим

$$C_{Pq}^{H-1} \leq \frac{(qP)^{H-1}}{(H-1)!} < \frac{H(qP)^{H-1} e^H}{H^H \sqrt{H}} < \frac{H(H^2)^{H-1} (P)^{H-1} e^H}{H^H \sqrt{H}} < P^{H-1} e^H H^{H-1}$$

различных значений q порядка H^2 . Таким образом, трудоёмкость всего алгоритма равна $O((2e)^H H^{H+3} P^{2H-1})$. Если H фиксировано, то трудоёмкость алгоритма составит $O(P^{2H-1})$, и он псевдополиномиален.

Теорема 2. *Задача минимизации длины цикла при условии, что число одновременно обрабатываемых деталей ограничено фиксированной величиной H , псевдополиномиально разрешима.*

Предложенный алгоритм может быть использован в случае ограничения на возобновимые ресурсы. Результаты работы могут быть полезны, если необходимо решить вопрос о необходимом количестве машин каждого типа.

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 деталей на основе циклических расписаний // Дискрет. анализ и исслед. операций. 2008. Т. 15, № 5. С. 47–60.
3. Сервах В. В. Эффективно разрешимый случай задачи календарного планирования с возобновимыми ресурсами // Дискрет. анализ и исслед. операций. Сер. 2. 2000. Т. 7, № 1. С. 75–82.
4. Тимковский В. Г. Приближённое решение задачи построения расписания для циклических систем // Эконом.-мат. методы. 1986. Т. 22, № 1. С. 171–174.
5. Boudoukh T., Penn M., Weiss G. Job-shop — an application of fluid approximation // Proc. 10th Conf. Ind. Eng. Management (Haifa, Israel, June 10–12, 1998). Haifa, Israel: Isr. Inst. Technol., 1998. P. 254–258.
6. Boudoukh T., Penn M., Weiss G. Scheduling jobshops with some identical or similar jobs // J. Sched. 2001. Vol. 4, No. 4. P. 177–199.
7. Brucker P. Scheduling algorithms. Berlin; Heidelberg; New York: Springer-Verl., 2007. 371 p.
8. Hall N. G., Lee T. E., Posner M. E. The complexity of cyclic shop scheduling problems // J. Sched. 2002. Vol. 5, No. 4. P. 307–327.
9. Hanen C. Study of a NP-hard cyclic scheduling problem: The recurrent job-shop // Eur. J. Oper. Res. 1994. Vol. 72, No. 1. P. 82–101.
10. Kamoun H., Sriskandarajah C. The complexity of scheduling jobs in repetitive manufacturing systems // Eur. J. Oper. Res. 1993. Vol. 70, No. 3. P. 350–364.
11. Levner E., Kats V., Pablo D., Cheng E. Complexity of cyclic scheduling problems: A state-of-the-art survey // Comput. Ind. Eng. 2010. Vol. 59, No. 2. P. 352–361.
12. McCormick S. T., Rao U. S. Some complexity results in cyclic scheduling // Math. Comput. Model. 1994. Vol. 20, No. 2. P. 107–122.
13. Rao U. S., Jackson P. L. Subproblems in identical jobs cyclic scheduling: properties, complexity, and solution approaches. Tech. Rep. Ithaca, NY: Cornell Univ. Press, 1993. 47 p.
14. Roundy R. Cyclic schedules for job shops with identical jobs // Math. Oper. Res. 1992. Vol. 17, No. 4. P. 842–865.

15. **Servakh V. V.** A dynamic algorithm for some project management problems // Proc. Int. Workshop Discrete Optimization Methods in Scheduling and Computer-Aided Design (Minsk, Sept. 5–6, 2000). Minsk: Inst. Eng. Cybern. NAS Belarus, 2000. P. 90–92.
16. **Timkovsky V. G.** Cycle shop scheduling // Handbook of scheduling: algorithms, models, and performance analysis (J. Y.-T. Leung, ed.). London: Chapman & Hall/CRC, 2004. P. 127–148.

*Боброва Екатерина Александровна,
Сервах Владимир Вицентьевич*

Статья поступила
3 июля 2015 г.

Исправленный вариант —
30 августа 2016 г.

UDC 519.8

DOI: 10.17377/daio.2017.24.500

CONSTRUCTION OF CYCLIC SCHEDULES
IN PRESENCE OF PARALLEL MACHINES

E. A. Bobrova^a, V. V. Servakh^b

Omsk Branch of Sobolev Institute of Mathematics,
13 Pevtsov St., 644043 Omsk, Russia,

E-mail: ^aeabobrova88@gmail.com, ^bsvv_usa@rambler.ru

Abstract. We consider the problem of processing some identical jobs with a complicated technological route on some production line in presence of parallel machines. Under some constraints on the number of jobs processed simultaneously, a cyclic schedule is desired with minimum cycle duration. Some algorithm for construction of an exact solution is proposed and substantiated. Also, we found the case of pseudopolynomially solvable problem. Illustr. 4, bibliogr. 16.

Keywords: cyclic schedule, dynamic programming, pseudopolynomial algorithm.

REFERENCES

1. **E. A. Bobrova, A. A. Romanova, and V. V. Servakh**, The complexity of cyclic scheduling for identical jobs, *Diskretn. Anal. Issled. Oper.*, **20**, No. 4, 3–14, 2013 [Russian].
2. **A. A. Romanova and V. V. Servakh**, Optimization of processing identical jobs by means of cyclic schedules, *Diskretn. Anal. Issled. Oper.*, **15**, No. 5, 47–60, 2008 [Russian]. Translated in *J. Appl. Ind. Math.*, **3**, No. 4, 496–504, 2009.
3. **V. V. Servakh**, An effectively solvable case of a project scheduling problem with renewable resources, *Diskretn. Anal. Issled. Oper., Ser. 2*, **7**, No. 1, 75–82, 2000 [Russian].
4. **V. G. Timkovsky**, Approximate solution of schedule construction problem for cyclic system, *Ekonom. Mat. Metody*, **22**, No. 1, 171–174, 1986 [Russian].
5. **T. Boudoukh, M. Penn, and G. Weiss**, Job-shop — an application of fluid approximation, in *Proc. 10th Conf. Ind. Eng. Manag., Haifa, Israel, June 10–12, 1998*, pp. 254–258, Isr. Inst. Technol., Haifa, 1998 [Hebrew].
6. **T. Boudoukh, M. Penn, and G. Weiss**, Scheduling jobshops with some identical or similar jobs, *J. Sched.*, **4**, No. 4, 177–199, 2001.
7. **P. Brucker**, *Scheduling Algorithms*, Springer, Heidelberg, 2007.

8. **N. G. Hall, T. E. Lee, and M. E. Posner**, The complexity of cyclic shop scheduling problems, *J. Sched.*, **5**, No. 4, 307–327, 2002.
9. **C. Hanen**, Study of a NP-hard cyclic scheduling problem: The recurrent job-shop, *Eur. J. Oper. Res.*, **72**, No. 1, 82–101, 1994.
10. **H. Kamoun and C. Sriskandarajah**, The complexity of scheduling jobs in repetitive manufacturing systems, *Eur. J. Oper. Res.*, **70**, No. 3, 350–364, 1993.
11. **E. Levner, V. Kats, D. Pablo, and E. Cheng**, Complexity of cyclic scheduling problems: A state-of-the-art survey, *Comput. Ind. Eng.*, **59**, No. 2, 352–361, 2010.
12. **S. T. McCormick and U. S. Rao**, Some complexity results in cyclic scheduling, *Math. Comput. Model.*, **20**, No. 2, 107–122, 1994.
13. **U. S. Rao and P. L. Jackson**, Subproblems in identical jobs cyclic scheduling: Properties, complexity and solution approaches, *Tech. Rep.*, Cornell Univ., Ithaca, NY, USA, 1993. Available at <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.45.3814>. Accessed Oct. 5, 2016.
14. **R. Roundy**, Cyclic schedules for job shops with identical jobs, *Math. Oper. Res.*, **17**, No. 4, 842–865, 1992.
15. **V. V. Servakh**, A dynamic algorithm for some project management problems, in *Proc. Int. Workshop “Discrete Optimization Methods in Scheduling and Computer-Aided Design”*, Minsk, Belarus, Sept. 5–6, pp. 90–92, Inst. Eng. Cybern. NAS Belarus, Minsk, 2000.
16. **V. G. Timkovsky**, Cycle shop scheduling, in *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, pp. 127–148, CRC Press, Boca Raton, 2004.

Ekaterina A. Bobrova,
Vladimir V. Servakh

Received
3 July 2015
Revised
30 August 2016