

УДК 519.2+621.391

ОПТИМИЗАЦИЯ ВЫПУСКА ОДНОТИПНЫХ ДЕТАЛЕЙ НА ОСНОВЕ ЦИКЛИЧЕСКИХ РАСПИСАНИЙ

А. А. Романова, В. В. Сервах

Аннотация. Исследуются задачи составления циклических расписаний при выпуске однотипных деталей в гибких производственных системах. Предложен алгоритм точного решения для одной из таких задач, основанный на методе динамического программирования. Построена вполне полиномиальная аппроксимационная схема для задачи минимизации циклического времени в случае, когда число деталей, одновременно находящихся в процессе обработки, ограничено константой.

Ключевые слова: циклическое расписание, идентичные детали, динамическое программирование, аппроксимационная схема.

1. Постановка задачи

Циклические расписания играют важную роль в организации производства. Они обеспечивают ритмичность загрузки оборудования и исполнителей, позволяют эффективно планировать как поставки ресурсов, так и сбыт продукции. Сложность задач построения циклических расписаний с различными технологическими ограничениями исследовалась в [7, 8, 11–13, 16]. Большинство рассматриваемых задач являются NP-трудными в сильном смысле [8, 16], поэтому целесообразным является построение приближённых и эвристических алгоритмов их решения. Такие алгоритмы предлагаются в [3, 5, 6, 14]. С другой стороны, в [8, 16] разработаны точные алгоритмы решения задач, основанные на методе ветвей и границ. В настоящей работе основное внимание будет уделено задаче минимизации циклического времени в случае ограниченного числа деталей, участвующих в цикле. Ранее для этой задачи был предложен метод ветвей и границ [8].

Рассмотрим постановку задачи. На производственной линии, состоящей из m различных машин, необходимо обработать партию однотипных деталей. Все детали проходят одинаковый технологический маршрут (O_1, O_2, \dots, O_n) обработки, состоящий из n последовательно выполняемых операций. Операция O_j выполняется на машине m_j в течение

p_j единиц времени, $j = 1, \dots, n$. Машины в технологическом маршруте могут повторяться. Прерывания операций запрещены. Одновременное выполнение двух и более операций на одной машине не допускается.

Обозначим через $t(j; k)$ время начала выполнения операции O_j детали k . Расписание называется *циклическим*, если для любого $j = 1, \dots, n$ и для любого $k \in Z$ выполняется $t(j; k) = t_j + Ck$, где $t_j = t(j; 0)$ и C — *длина цикла*, или *циклическое время*. Циклическое расписание определяется заданием времени цикла C и времени t_j начала выполнения каждой операции $j = 1, \dots, n$ для одной из деталей. Наряду с временем цикла C важной характеристикой циклического расписания является величина h — максимальное число одновременно обрабатываемых деталей. При этом считаем, что деталь находится в обработке, если выполнение её операций уже началось, но последняя операция ещё не завершилась.

Циклические расписания часто используются в производственных системах в силу их простоты и удобства применения, поэтому разработка алгоритмов построения циклических расписаний является актуальным направлением исследований в области теории расписаний. Минимизация времени цикла при большом количестве деталей обеспечивает максимальную производительность линии. Задача построения циклического расписания с минимальным временем цикла является полиномиально разрешимой. Однако, если число одновременно обрабатываемых деталей ограничено и это ограничение является параметром, зависящим от входа задачи, то эта задача становится NP-трудной в сильном смысле [8].

Примером производственной линии с таким ограничением является сборка самолётов, когда количество позиций для сборки ограничено. Другим примером может служить химическая обработка деталей. Деталь крепится на оснастку и последовательно обрабатывается в различных резервуарах. Количество одновременно обрабатываемых деталей ограничено числом таких оснасток, которое также фиксировано. Таким образом, параметр h является важной характеристикой циклических расписаний.

Далее будем рассматривать задачу минимизации времени цикла при условии, что максимальное количество h деталей, одновременно находящихся в обработке, ограничено: $h \leq H$.

При $H \geq n$ задача полиномиально разрешима. Минимальное время цикла C^* при этом равно суммарному времени выполнения всех операций одной детали на самой загруженной машине. Можно использовать следующий простой алгоритм. Первую операцию начинаем в момент начала цикла. Если вторая операция использует другую машину, то её вы-

полняем 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 отношение предшествования операций одной детали, то передвигаем выполнение текущей операции вперёд на время, равное C^* , а в текущем цикле переходим к детали с меньшим номером. В худшем случае в каждом цикле выполняется по одной операции, поэтому оптимальное расписание для задачи при $H \geq n$ можно получить, просто применив описанный алгоритм.

Рассматриваемая задача также полиномиально разрешима, когда все операции детали обрабатываются на различных машинах. В этом случае минимальная длина цикла равна

$$\max\{P/H, p_{\max}\},$$

$$\text{где } P = \sum_{j=1}^n p_j, p_{\max} = \max_{j=1, \dots, n} p_j.$$

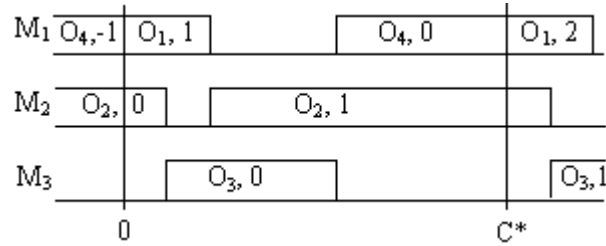
Изложим идею алгоритма решения задачи в общем случае. В промежутке времени, равном длине цикла, выполняется весь набор операций. При этом операции могут относиться к разным деталям и таких деталей не должно быть больше чем H . Если во время цикла выполняются операции O_j и O_k одной детали, то в силу ограничений предшествования все операции $O_{j+1}, O_{j+2}, \dots, O_{k-2}, O_{k-1}$ этой детали должны быть выполнены в этом же цикле. Таким образом, технологический маршрут можно разделить на цепи операций, которые относятся к разным деталям и выполняются в одном цикле. При заданном разбиении на цепи рассматриваемая задача сводится к разномаршрутной задаче job shop с H требованиями и m приборами. Технологический маршрут обработки требования k соответствует цепи k в разбиении, $k = 1, \dots, H$. Необходимо составить расписание обслуживания требований за наименьшее время. Алгоритм решения задачи заключается в решении для каждого разбиения задачи job shop и выборе разбиения с минимальным значением длины цикла.

В разд. 2 устанавливаются свойства оптимальных решений задачи, позволяющие оценить число возможных разбиений на цепи. В разд. 3 предлагается алгоритм динамического программирования для вспомогательной разномаршрутной задачи и приводится общая схема алгоритма решения задачи. В разд. 4 предлагается вполне полиномиальная аппроксимационная схема для важного частного случая рассматриваемой задачи.

2. Свойства оптимальных расписаний

В данном разделе устанавливаются свойства оптимальных расписаний, связанные с разбиением технологического маршрута на цепи. Отрезок $[0, P]$ разобьём на H частей точками $0 = \xi_0 < \xi_1 < \dots < \xi_H = P$ и будем говорить, что произведено разбиение $R_H = (\xi_1, \xi_2, \dots, \xi_{H-1})$ технологического маршрута на цепи. К сожалению, нельзя обойтись разбиениями, в которых точки ξ_i целочисленны.

Рассмотрим пример. Пусть $n = 4$, $m = 3$, $H = 2$. Обработка детали заключается в последовательном выполнении 4 операций с длительностями $(1, 4, 2, 2)$ на машинах $(1, 2, 3, 1)$ соответственно. Оптимальная длина цикла при этом равна $C^* = 4, 5$; $\xi_1 = 4, 5$. Соответствующее циклическое расписание выглядит следующим образом:



Таким образом, возможных разбиений может быть бесконечно много. Докажем, что для поиска оптимального решения задачи достаточно рассмотреть конечное число разбиений.

Обозначим суммарную длительность первых i операций через $T_i = \sum_{j=1}^i p_j$, $i = 1, 2, \dots, n$. Пусть $T = \{0, T_1, T_2, \dots, T_n\}$. Если $\xi_j \in T$, то разрез происходит между операциями. Если $\xi_j \notin T$, то разрез между цепями j и $j + 1$ происходит внутри операции. В этом случае в силу непрерывности выполнения операций части разрезанной операции должны примыкать к границам цикла. Причём первая часть примыкает к концу цикла, а последняя — к началу. Отсюда непосредственно вытекает, что каждый интервал $(T_j; T_{j+1})$ содержит не более одной точки разбиения, так как иначе на некоторой машине будет выполняться более одной операции одновременно.

Рассмотрим разбиения $R_H = (\xi_1, \xi_2, \dots, \xi_{H-1})$, где каждая точка ξ_j либо является элементом множества T , либо принадлежит интервалу (T_{i_j}, T_{i_j+1}) . Пусть k точек разбиения лежат в интервалах (T_{i_j}, T_{i_j+1}) , а $H - k - 1$ точек принадлежат множеству T . Обозначим через I^{dec} множество номеров операций, внутри которых происходит разрез. Заметим,

что имеет смысл рассматривать только такие множества I^{dec} , что операции этого множества выполняются на попарно различных машинах. Каждая операция $O_j \in I^{\text{dec}}$ разбивается на две части O_j^l и O_j^r , $j \in I^{\text{dec}}$. Пусть $I = \{1, 2, \dots, n\} \setminus I^{\text{dec}}$ — множество номеров остальных операций. Выделим в I подмножество I^{last} номеров конечных операций цепей.

Рассмотрим граф предшествования $G = (V, E)$. Здесь

$$V = \{O_j^l \cup O_j^r, j \in I^{\text{dec}}\} \cup \{O_j, j \in I\},$$

а E — множество дуг, задающих ограничения предшествования в цепях и порядок на множестве операций, выполняемых на каждой из машин. Очевидно, что нужно рассматривать только такие графы, в которых операции O_j^r стоят первыми в своей цепи и на своей машине, а операции O_j^l стоят последними в своей цепи и на своей машине. Необходимо найти расписание, в котором время окончания выполнения самой поздней операции минимально. Для исследования свойств такого расписания при фиксированном порядке выполнения работ на каждой машине составим задачу линейного программирования.

Опишем переменные задачи:

s_j — время начала выполнения операции O_j , $j \in I$, или операции O_j^l , $j \in I^{\text{dec}}$;

x_j — длительность операции O_j^l , $x_j \leq p_j$, $j \in I^{\text{dec}}$;

C_{max} — общее время завершения операций всех цепей.

Операция O_j^r имеет длительность $p_j - x_j$ и начинается в нулевой момент времени, а операция O_j^l завершается в момент времени C_{max} , $j \in I^{\text{dec}}$.

Разобьём множество E на 4 подмножества:

- 1) $E_1 = \{(O_i, O_j) \in E \mid i \in I, j \in I\}$;
- 2) $E_2 = \{(O_i, O_j^l) \in E \mid i \in I, j \in I^{\text{dec}}\}$;
- 3) $E_3 = \{(O_i^r, O_j) \in E \mid i \in I^{\text{dec}}, j \in I\}$;
- 4) $E_4 = \{(O_i^r, O_j^l) \in E \mid i \in I^{\text{dec}}, j \in I^{\text{dec}}\}$.

С каждой дугой из этих множеств свяжем ограничения предшествования:

$$s_i + p_i \leq s_j, (O_i, O_j) \in E_1, (O_i, O_j^l) \in E_2,$$

$$p_i - x_i \leq s_j, (O_i^r, O_j) \in E_3, (O_i^r, O_j^l) \in E_4.$$

Конечные операции цепей необходимо завершить не позднее момента C_{max} :

$$s_j + p_j \leq C_{\text{max}}, j \in I^{\text{last}};$$

$$s_j + x_j = C_{\max}, \quad j \in I^{\text{dec}}.$$

Таким образом, имеем модель задачи:

$$\begin{aligned} C_{\max} &\rightarrow \min, \\ C_{\max} - s_j &\geq p_j, & j \in I^{\text{last}}, \\ C_{\max} - s_j - x_j &= 0, & j \in I^{\text{dec}}, \\ s_j - s_i &\geq p_i, & (O_i, O_j) \in E_1, (O_i, O_j^l) \in E_2, \\ s_j + x_i &\geq p_i, & (O_i^r, O_j) \in E_3, (O_i^r, O_j^l) \in E_4, \\ -x_j &\geq -p_j, & j \in I^{\text{dec}}, \\ s_i &\geq 0, & i \in I \cup I^{\text{dec}}, \\ x_j &\geq 0, & j \in I^{\text{dec}}. \end{aligned} \tag{1}$$

Пусть A — матрица ограничений задачи (1), b — вектор правых частей ограничений, $y = (s_1, s_2, \dots, s_n, x_{j_1}, x_{j_2}, \dots, x_{j_k}, C_{\max})$ — вектор переменных задачи, где $j_i \in I^{\text{dec}}$, $i = 1, \dots, k$.

Лемма 1. *Любой минор матрицы A по модулю не превосходит H .*

ДОКАЗАТЕЛЬСТВО. Рассмотрим произвольный ненулевой минор M матрицы A . Пусть L — его порядок, а B — соответствующая матрица. Будем считать, что в матрице B строки и столбцы идут в том же порядке, что и в матрице A . Докажем, что значение минора M по модулю не превышает H . В последнем столбце матрицы A стоит не более H единиц, так как переменная C_{\max} появляется в ограничениях на конечные операции цепей, а цепей не более H . Матрица B может содержать элементы последнего столбца матрицы A , а может их и не содержать. Рассмотрим первый случай. Тогда в последнем столбце матрицы B не более H единиц, причём сначала идут единицы, а потом нули. Пусть единиц в последнем столбце p штук ($p \leq H$). Вычислим величину минора M . Рассмотрим его разложение по последнему столбцу $M = \sum_{i=1}^p (-1)^{i+L} M_{iL}$. Здесь M_{iL} — определитель матрицы, полученной из B вычеркиванием строки i и столбца L . Разобьём такую матрицу на блоки. Столбцы разделим на 2 типа: S_{i_v} , соответствующие переменным s_{i_v} , $v = 1, \dots, L_1$, и X_{j_u} , соответствующие x_{j_u} , $u = 1, \dots, L - L_1 - 1$, а строки группируем в соответствии с моделью (1). Матрица будет иметь вид

S_{i_v}	X_{j_u}
B_1	0
B_2	B_3
B_4	0
B_5	B_6
0	B_7

Здесь 0 означает блок матрицы, состоящий из нулей. Первые два блока по строкам соответствуют первому типу ограничений задачи, в которых перед некоторой переменной s_j стоит -1 , поэтому в блоке B_1 в каждой строке не более одной -1 . В блоках B_2 и B_3 в каждой строке не более одной -1 , так как им соответствует второй тип ограничений. В неравенствах третьего типа ограничений перед некоторой переменной s_j стоит 1, а перед некоторой переменной s_i стоит -1 . Поэтому в блоке B_4 в каждой строке не более одной 1 и не более одной -1 . Следующий блок матрицы состоит из нулей. В ограничениях четвертого типа какая-то переменная s_j и какая-то переменная x_i встречаются с коэффициентом 1, поэтому в строках блоков B_5 и B_6 не более одной 1. Наконец, последнему типу ограничений соответствуют блок из нулей и блок B_7 , в котором в каждой строке не более одной -1 . Отметим, что в матрице, соответствующей минору M_{iL} , в каждой строке не более двух ненулевых элементов.

Посчитаем значение минора M_{iL} . Последовательно разложим его по строкам и столбцам, в которых только один ненулевой элемент, т. е. 1 или -1 . В итоге останется либо одно из чисел 1, 0, или -1 , либо матрица, в которой нет строк и столбцов с одним ненулевым элементом. Рассмотрим случай, когда порядок оставшейся матрицы равен $l > 1$. Заметим, что эта матрица может содержать только строки, находящиеся в блоках B_2, B_3, B_4, B_5, B_6 . Если столбцов из блоков B_3 и B_6 в матрице не оказалось, то по строкам, находящимся в блоках B_2 и B_5 , определитель раскладывается дальше. Тогда в матрице останутся только строки и столбцы из блока B_4 . Напомним, что в каждой строке данного блока ровно одна 1 и одна -1 , поэтому при суммировании всех столбцов этой матрицы получается нулевой столбец. Если столбцы из блоков B_3 и B_6 в матрице присутствуют, то и в этом случае имеем нулевой определитель. Действительно, рассмотрим следующую линейную комбинацию столбцов: $\sum_{v=1}^{l_1} S_{i_v} - \sum_{u=1}^{l-l_1} X_{j_u}$. Из вида матрицы ясно, что получится нулевой столбец, а значит, столбцы матрицы линейно зависимы. Таким образом, каждый минор M_{iL} равен либо 0, либо -1 , либо 1. Так как в разложении минора M число слагаемых равно $p \leq H$, то он принимает целые значения из промежутка $[-H; H]$.

Второй вариант, когда матрица B не содержит последний столбец матрицы A , сводится к уже рассмотренному случаю. Тогда минор M совпадает по виду с минором M_{iL} , а его значение равно $-1, 0$ или 1 . Лемма 1 доказана.

Теорема 1. Для задачи минимизации длины цикла при условии, что

число одновременно обрабатываемых деталей не превосходит H , существуют оптимальное расписание $\{t_j\}$, $j = 1, \dots, n$, и целое число $q \leq H$ такие, что все t_j и минимальное время цикла C^* кратны дроби $1/q$.

ДОКАЗАТЕЛЬСТВО. Существует оптимальное решение задачи (1), которому соответствует некоторая вершина многогранника допустимых решений. Значит, оно является решением некоторой системы $B\bar{y} = \bar{b}$, где B — матрица, соответствующая ненулевому минору целочисленной матрицы A , \bar{b} — целочисленный вектор, \bar{y} — вектор базисных переменных. Тогда знаменатель компонент вектора $\bar{y} = B^{-1}\bar{b}$ не может быть больше H , так как $|B| \leq H$ по лемме 1. Причём, так как все компоненты вектора \bar{y} получаются делением на один и тот же минор $|B|$, то они, в том числе и $C^* = C_{\max}$, кратны дроби $1/|B|$. Заметим, что расписание $\{t_j\}$, $j = 1, \dots, n$, легко вычисляется через переменные задачи (1), т. е. через компоненты вектора \bar{y} . А именно, $t_j = s_j + (k-1)C^*$, где k — номер цепи, в которой оказалась операция O_j (или O_j^l) при фиксировании разбиения. Так как s_j и C^* кратны дроби $1/|B|$, то и величина t_j также кратна этой дроби. Теорема 1 доказана.

К сожалению, данный подход не приводит к эффективному методу решения исходной задачи, так как приходится перебирать все последовательности выполнения операций на каждой машине. Ниже предлагается алгоритм решения задачи, основанный на схеме динамического программирования.

3. Алгоритм решения задачи минимизации времени цикла с ограничением на число одновременно обрабатываемых деталей

Рассмотрим сначала алгоритм решения задачи при фиксированном разбиении на цепи. В силу теоремы 1 достаточно рассматривать разбиения с точками, принадлежащими одному из множеств $A_q = \{r_i/q \mid r_i = 1, \dots, qP\}$, $q = 1, \dots, H$. Для каждого $q = 1, \dots, H$ определим множество \mathcal{R}_q допустимых разбиений. Разбиение $R_H = (\xi_1, \xi_2, \dots, \xi_{H-1})$, где $\xi_i \in A_q$, $i = 1, 2, \dots, H-1$, принадлежит множеству \mathcal{R}_q при выполнении следующих условий:

- 1) каждый интервал $(T_j; T_{j+1})$ содержит не более одной точки ξ_i ,
- 2) если $\xi_u \in (T_{j_u}; T_{j_u+1})$ и $\xi_v \in (T_{j_v}; T_{j_v+1})$, то операции O_{j_u} и O_{j_v} выполняются на разных машинах.

Технологический маршрут разбивается на H цепей. Если с каждой из цепей связать требование, а с машиной — прибор, то получим задачу обслуживания требований с разными технологическими маршрутами

(job shop) и дополнительным ограничением непрерывного выполнения операций, внутри которых производится разрез. Если точка разбиения принадлежит интервалу $(T_i; T_{i+1})$, то операция O_{i+1} заменяется на две O_{i+1}^l и O_{i+1}^r . Причём момент окончания операции O_{i+1}^l должен совпадать с моментом окончания цикла, а операция O_{i+1}^r должна начинаться в момент начала цикла. Так как точки разбиения могут быть нецелочисленными, то длительности некоторых операций в задаче job shop могут получиться нецелыми. Для дальнейшего изложения нам важна целочисленность рассматриваемых операций. Поэтому, учитывая результат теоремы 1, необходимо решить H задач job shop с длительностями операции $\hat{p}_j = p_j q$ при $q = 1, \dots, H$.

Для решения целочисленной задачи job shop был предложен алгоритм динамического программирования, основная идея которого изложена в [2, 17] и заключается в переборе промежуточных состояний обслуживания требований и поиске для каждого состояния лучшего частичного расписания.

Через S_k обозначим суммарную длительность операций требования k , $k = 1, 2, \dots, H$. Текущее состояние выполнения операций требования k будем задавать величиной x_k . Если $x_k = 0$, то операции этого требования ещё не начали выполняться. Если $x_k = S_k$, то все операции этого требования уже завершены. Промежуточное значение x_k означает, что общее время выполнения операций этого требования, без учёта простоев, составило x_k и требуется ещё $S_k - x_k$ единиц времени для их завершения. Вектор $\mathbf{x} = (x_1, x_2, \dots, x_H)$ называется *состоянием* выполнения требований, $\mathbf{0} = (0, 0, \dots, 0)$ — начальное состояние, т. е. ни одна операция не начала своего выполнения, $\mathbf{S} = (S_1, S_2, \dots, S_H)$ — конечное состояние, т. е. операции всех требований выполнены.

При целых значениях \hat{p}_j можно ограничиться анализом только целочисленных моментов времени и, как следствие, рассматривать только целочисленные векторы \mathbf{x} . Через

$$X = \{\mathbf{x} = (x_1, x_2, \dots, x_H) \mid x_k \in \{0, 1, \dots, S_k\}, k = 1, 2, \dots, H\}$$

обозначим множество всех состояний.

Переход из одного состояния в другое осуществляется с помощью соответствующего управления. В рассматриваемом случае, *управления* суть булевы векторы $\Delta = (\delta_1, \delta_2, \dots, \delta_H)$, где $\delta_k \in \{0, 1\}$, $k = 1, 2, \dots, H$. При этом из состояния \mathbf{x} переходим в состояние $\mathbf{x} + \Delta$. Процесс перехода соответствует одновременному выполнению в единичном временном интервале операций тех требований, для которых $\delta_k = 1$.

Если состояние \mathbf{x} соответствует тому, что операция некоторого требования k была начата и ещё не завершена, то в силу непрерывности выполнения операций $\delta_k = 1$. Тогда переход $\mathbf{x} \rightarrow \mathbf{x} + \Delta$, у которого $\delta_k = 0$, будет недопустим.

Переход $\mathbf{x} \rightarrow \mathbf{x} + \Delta$ однозначно определяет множество выполняемых операций. Если какая-либо пара из них выполняется на одной машине, то управление Δ запрещается.

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 обслуживание требования k начинается с выполнения операции O_i^r . Аналогично для всех допустимых управлений Δ , переводящих систему в состояние \mathbf{S} , должно быть выполнено $\delta_k = 1$, если обслуживание требования k заканчивается выполнением операции O_j^l .

Множество допустимых управлений, приводящих в состояние \mathbf{x} , обозначим через $\Delta_{\mathbf{x}}$. В задаче необходимо найти последовательность допустимых управлений, которая за минимальное число переходов позволяет перейти из состояния $\mathbf{0}$ в состояние \mathbf{S} . Пусть $L(\mathbf{x})$ — наименьшее число переходов из состояния $\mathbf{0}$ в состояние \mathbf{x} . Выпишем рекуррентное соотношение

$$L(\mathbf{0}) = 0;$$

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

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.

Оценим время работы алгоритма. Для каждого состояния $\mathbf{x} \in X$ необходимо вычислить $L(\mathbf{x})$, что требует перебора $2^H - 1$ булевых векторов Δ . При этом проверка допустимости вектора требует порядка $O(H^2)$ операций. Всего состояний $(S_1 + 1)(S_2 + 1) \dots (S_H + 1)$. В итоге общая трудоёмкость алгоритма составит $O(2^H H^2 S_1 S_2 \dots S_H)$ операций.

При фиксированном разбиении и фиксированном q оценим трудоём-

кость решения соответствующей задачи job shop. Заметим, что

$$S_1 S_2 \dots S_H \leq (qP/H)^H,$$

так как $\sum_{k=1}^H S_k = qP$. Очевидно, $qP/H \leq P$, поэтому решение задачи job shop требует не более $O(2^H H^2 P^H)$ операций.

Опишем алгоритм решения исходной задачи.

АЛГОРИТМ СН.

Полагаем $C^* = P$, $t_j = 0$, $j = 1, \dots, n$.

ЦИКЛ от $q = 1$ до H :

ЦИКЛ по всем $R_H \in \mathcal{R}_q$:

Решить соответствующую задачу job shop с длительностями операций $\hat{p}_j = qp_j$ (если O_j при разбиении разбилась на две операции O_j^l и O_j^r , то обозначим через p_j^l и p_j^r длительности операций O_j^l и O_j^r соответственно; в этом случае $\hat{p}_j^l = qp_j^l$, $\hat{p}_j^r = qp_j^r$).

Пусть s_j — время начала выполнения операции O_j или O_j^l , $j = 1, \dots, n$, в оптимальном расписании задачи job shop, $C = \max_{j=1, \dots, n} (s_j + \hat{p}_j)$.

Если $C/q < C^*$, то полагаем $C^* = C/q$, $t_j = s_j/q + (k-1)C^*$, $j = 1, \dots, n$, где k — номер цепи, в которой оказалась операция O_j (или O_j^l) при разбиении.

КОНЕЦ.

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

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

Трудоёмкость итерации q , $q = 1, \dots, H$, не превосходит

$$O((2e)^H H^{3/2} P^{2H-1}).$$

Таким образом, алгоритм С решает рассматриваемую задачу за время $O((2e)^H H^{5/2} P^{2H-1})$. Если H фиксировано, то трудоёмкость алгоритма составит $O(P^{2H-1})$.

Теорема 2. Задача минимизации времени цикла при условии, что число одновременно обрабатываемых деталей ограничено фиксированной величиной H , является псевдополиномиально разрешимой.

4. Аппроксимационная схема

Под ρ -приближённым алгоритмом решения задачи на минимум будем понимать алгоритм, находящий приближённое решение, целевая функция которого не более чем в ρ раз превышает целевую функцию оптимального решения (если задача разрешима). Соответствующее решение будем называть ρ -приближённым решением.

Под вполне полиномиальной аппроксимационной схемой (FPTAS) будем понимать семейство $(1 + \varepsilon)$ -приближённых алгоритмов при всевозможных $\varepsilon > 0$ с временной сложностью, полиномиально зависящей от длины входа задачи и от $1/\varepsilon$.

В данном разделе предлагается вполне полиномиальная аппроксимационная схема решения задачи минимизации времени цикла в случае, когда число одновременно обрабатываемых деталей ограничено константой H .

Теорема 3. Для нахождения $(1 + \varepsilon)$ -приближённого решения в задаче минимизации времени цикла при ограничении числа одновременно обрабатываемых деталей фиксированной величиной H существует вполне полиномиальная аппроксимационная схема трудоёмкости $O\left(\left(\frac{n}{\varepsilon}\right)^{2H-1}\right)$.

ДОКАЗАТЕЛЬСТВО. Применим подход, основанный на округлении входных данных (см., например, [1, 10]). Фиксируем $\varepsilon > 0$. Выберем коэффициент округления $\delta = \frac{\varepsilon P}{nH}$ и рассмотрим упрощённую задачу с длительностями $\bar{p}_j = \lceil p_j / \delta \rceil$. Обозначим через ОРТ и $\overline{\text{ОРТ}}$ оптимальные значения целевой функции исходной и упрощённой задач соответственно. Заметим, что $\bar{p}_j \leq p_j / \delta + 1$, поэтому

$$\bar{P} \leq \sum_{j=1}^n \frac{p_j}{\delta} + n = \frac{P}{\delta} + n = \frac{nH}{\varepsilon} + n.$$

Тогда

$$\begin{aligned} \bar{P}^{2H-1} &\leq \left(\frac{nH}{\varepsilon}\right)^{2H-1} \left(1 + \frac{\varepsilon}{H}\right)^{2H-1} \leq \left(\frac{nH}{\varepsilon}\right)^{2H-1} (1 + 2\varepsilon)^2 \\ &\leq O\left(\left(\frac{nH}{\varepsilon}\right)^{2H-1}\right). \end{aligned}$$

Вычислим трудоёмкость решения упрощённой задачи алгоритмом СН. Он решает задачу за время

$$O((2e)^H H^{5/2} \bar{P}^{2H-1}) \leq O((2e)^H H^{2H+1,5} (n/\varepsilon)^{2H-1}).$$

С учётом того, что H является константой, трудоёмкость решения упрощённой задачи составляет $O((n/\varepsilon)^{2H-1})$ операций.

Для решения задачи достаточно найти разбиение на цепи и времена s_j начала операций в соответствующей задаче job shop. Пусть \bar{s}_j — время начала выполнения операции O_j в упрощённой задаче job shop, тогда приближённое решение исходной задачи зададим величинами $s_j = \delta \bar{s}_j$. Очевидно, оно остаётся допустимым, так как $p_j \leq \bar{p}_j \delta$. Если обозначить через L_i время окончания выполнения операций на машине i в приближённом решении, то для любого $i = 1, \dots, m$ его можно оценить следующим образом:

$$\begin{aligned} L_i &\leq \delta \bar{L}_i \leq \delta \overline{\text{OPT}} \leq \delta n + \text{OPT} \leq \varepsilon \frac{P}{H} + \text{OPT} \leq \varepsilon \text{OPT} + \text{OPT} \\ &= \text{OPT}(1 + \varepsilon). \end{aligned}$$

Поясним неравенство $\delta \overline{\text{OPT}} \leq \delta n + \text{OPT}$. Рассмотрим оптимальное расписание исходной задачи. Заменим в этом расписании каждую операцию с длительностью p_j операцией с длительностью $\delta \bar{p}_j$, сдвинув выполнение операций так, чтобы оставить расписание допустимым. Длина расписания увеличилась не более чем на δn . С другой стороны, $\delta \overline{\text{OPT}}$ — минимальная длина расписания с длительностями $\delta \bar{p}_j$, поэтому $\delta \overline{\text{OPT}} \leq \delta n + \text{OPT}$. Так как $L_i \leq \text{OPT}(1 + \varepsilon)$ для любого i , то длина расписания в приближённом решении равна $\max_{i=1, \dots, m} L_i \leq (1 + \varepsilon) \text{OPT}$. Теорема 3 доказана.

Авторы благодарят рецензента за полезные замечания.

ЛИТЕРАТУРА

1. Генс Г. В., Левнер Е. В. Эффективные приближённые алгоритмы для комбинаторных задач. Препринт. — М.: ЦЭМИ, 1981. — 38 с.
2. 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.
3. Aldakhilallah K. A., Ramesh R. Cyclic scheduling heuristics for a re-entrant job shop manufacturing environment // Internat. J. Production Research. — 2001. — Vol. 39. — P. 2635–2675.
4. Akers S. E. A graphical approach to production scheduling problems // Operations Research. — 1956. — Vol. 4, N 2. — P. 244–245.
5. Boudoukh T., Penn M., Weiss G. Scheduling job shops with some identical or similar jobs // J. Scheduling. — 2001. — Vol. 4. — P. 177–199.
6. Brucker P., Kampmeyer T. Tabu search algorithms for cyclic machine scheduling problems. Preprint. — Osnabrueck, 2002. — 24 p.

7. **Hall N. G., Lee T. E., Posner M. E.** The complexity of cyclic shop scheduling problems // J. Scheduling. — 2002. — Vol. 5. — P. 307–327.
8. **Hanan C.** Study of a NP-hard cyclic scheduling problem: the recurrent job-shop // Europ. J. Operational Research. — 1994. — Vol. 71. — P. 82–101.
9. **Hardgrave W. W., Nemhauser G. L.** A geometric model and a graphical algorithm for a sequencing problem // Operations Research. — 1963. — Vol. 11, N 6. — P. 889–900.
10. **Ibarra O. H., Kim C. E.** Fast approximation algorithms for the knapsack and sum of subset problems // J. ACM. — 1975. — V. 22, N 4. — P. 463–468.
11. **Kamoun H., Sriskandarajah C.** The complexity of scheduling jobs in repetitive manufacturing systems // Europ. J. Operational Research. — 1993. — Vol. 70. — P. 350–364.
12. **Lee T., Posner M.** Performance measure and schedules in periodic job shop // Operations Research. — 1997. — Vol. 45. — P. 72–91.
13. **McCormick S. T., Rao U. S.** Some complexity results in cyclic scheduling // Mathematical and Computer Modelling. — 1994. — Vol. 20. — P. 107–122.
14. **Rao U., Jackson P.** Identical jobs cyclic scheduling: subproblems, properties, complexity and solution approaches. — Ithaca, NY: Cornell Univ. Press, 1993. — 47 p.
15. **Romanova A. A., Servakh V. V.** On some cyclic machine scheduling problem // Abstracts of the XVII European Conference of Combinatorial Optimization (ECCO'2005). — Minsk: United Institute of Informatics Problems of the National Academy of Sciences of Belarus, 2005. — P. 58–59.
16. **Roundy R.** Cyclic schedules for job shops with identical jobs // Mathematics of Operations Research. — 1992. — Vol. 17, N 4. — P. 842–865.
17. **Servakh V. V.** A dynamic algorithm for some project management problems // Proceedings of the International Workshop “Discrete optimization methods in scheduling and computer-aided design”. — Minsk: National Academy of Sciences of Belarus Institute of Engineering Cybernetics, 2005. — P. 90–92.

Романова Анна Анатольевна,
e-mail: romanova_ann@bk.ru
Сервах Владимир Вицентьевич,
e-mail: svv_usa@rambler.ru

Статья поступила
19 декабря 2007 г.
Переработанный вариант —
30 июля 2008 г.