АЛГОРИТМ РЕШЕНИЯ ОДНОЙ ЭКСТРЕМАЛЬНОЙ ЗАДАЧИ ТЕОРИИ РАСПИСАНИЙ

УДК 004

Геннадий Александрович Беркетов,
к.т.н., профессор, профессор кафедры автоматизированных систем обработки информации и управления, Московский государственный университет экономики, статистики и информатики (МЭСИ)
Тел.: (495) 442-61-11
Эл. почта: GABerketov@mesi.ru

В статье рассматривается оригинальный алгоритм решения обобщенной задачи теории расписаний, основанный на методе ветвей и границ. Задачи составления расписания выполнения комплекса работ (операций) при ограничениях на используемые ресурсы часто возникают при календарном планировании операций дискретного производства, оптимизации сетевых графиков реализации научных, экономических или технических проектов. Инструментарий решения подобных задач включается в системы поддержки принятия решения АСУ многих предприятий.
Эффективность предлагаемого алгоритма позволяет решать с его помощью характерные для практики задачи большой размерности.

Ключевые слова: *общая задача теории расписаний, календарное планирование работ, метод ветвей и границ, системы поддержки принятия решений.*

Gennady A. Berketov,
PhD in Technical Sciences, Professor, Department of Automated Systems of Information Processing and Management, Moscow State University of Economics, Statistics and Informatics
Tel.: (495) 442-61-11
E-mail: GABerketov@mesi.ru

ALGORITHM FOR SOLVING EXTREME SCHEDULING PROBLEMS

The article considers the original algorithm for solving the generalized problem of scheduling theory, based on the branch and bound method. Task scheduling perform works (operations) and restrictions on resources used often occur with scheduling discrete manufacturing operations, optimizing network implementation schedules of scientific, economic or technical projects. Tools to solve such problems are included in the decision support system ACS in many businesses. The effectiveness of the proposed algorithm allows solving with it specific for practice large-scale problems.

Keywords: *general problem of scheduling, scheduling of work, branch and bound method, decision support system.*

Пусть необходимо обслужить n требований. Обслуживание i-го требования заключается в проведении над ним комплекса из n_i упорядоченных операций; j-ю операцию над i-м требованием обозначим через (i, j), причем, если (i, j) – операция предшествует (i, l) – операции $((i, j) < (i, l))$, то $j < l$. Каждая (i, j) – операция при наличии соответствующих ресурсов выполняется за время τ_{ij}; время ее выполнения не зависит от порядка, в котором обслуживаются требования.

Все операции над требованиями должны быть выполнены без прерывания и нарушения упорядоченности, т.е.

$$f_{ij} = S_{ij} + \tau_{ij} \text{ и } S_{ij} \geq f_{ij} \text{ для } j < l.$$

Где S_{ij} и f_{ij} – моменты начала и окончания выполнения (i, j) – операции соответственно. При обслуживании требований используется m видов ресурсов типа мощности; величина ресурса k-ого вида, необходимая для выполнения (i, j) – операции, определяется величиной $R_{ij}^k \geq 0$.

Суммарное требование к ресурсу k-ого вида в каждый момент времени не должно превышать величины R^k – мощности ресурса.

Расписание $S = \{S_{ij}\}$ (или $f = \{f_{ij}\}$) обслуживания требований назовем допустимым, если оно удовлетворяет перечисленным условиям. Очевидно, знание моментов окончания операций f_{ij} и продолжительность операций τ_{ij} позволяет легко находить S_{ij} – моменты начала операций.

Рассматривается следующая задача: среди допустимых расписаний найти оптимальное по быстродействию, для которого время, затрачиваемое на обслуживание всего пакета требований, минимально.

Алгоритм, предлагаемый в данной работе для решения вышепоставленной задачи, использует идеи метода ветвей и границ [1–4]. Алгоритм не требует запоминания дерева вариантов: вся информация, необходимая для вычислений запоминается в виде частичного решения специального вида. Эта особенность алгоритма позволяет значительно понизить требования к объему необходимой памяти, что существенно при решении задач большой размерности.

Прежде чем дать описание алгоритма, введем некоторые понятия и операции, которые потребуются нам в дальнейшем.

Введем переменные

$$x_{ij}(t) = \begin{cases} 1, \text{ если } f_{ij} = t, \\ 0, \text{ если } f_{ij} \neq t. \end{cases}$$

Следует заметить, что индекс t играет здесь чисто формальную роль и введен для удобства описания алгоритма. Очевидно, что для восстановления расписания по переменным $x_{ij}(t)$ нет необходимости в знании значений всех переменных.

Пусть для части операций уже составлено расписание; такое расписание будем называть частичным. Частичное решение запоминается как последовательность записей вида

$$I, j, t, x_{ij}(t).$$

Некоторые записи могут быть помечены (в качестве метки используется символ *). Содержательный смысл этой операции заключается в том, что при неоптимальном продолжении частичных решений происходит возврат к прежним вариантам и ищется новое продолжение; метки играют при этом значительную роль. Ниже приводится пример записи частичного решения, в которой третий элемент помечен:

1, 1, 3, 1| 1, 2, 5, 0| 2, 1, 4, 1 *| 2, 2, 7, 1

При машинной реализации алгоритма частичное решение представляет собой массив переменных указанной структуры; в качестве метки используется какой-либо числовой код. Работа алгоритма заключается в поиске наиболее оптимального продолжения частичного решения. В ходе поиска решения переменным $x_{ij}(t)$ присваиваются те или иные значения. Заметим, что при присвоении значений некоторым из n переменных другие переменные не могут принимать произвольных значений: их значения определяются ограничениями, накладываемые на допустимые решения. Такие переменные будем называть зависимыми.

Введем теперь понятия множества конфликтных операций и дефицитности ресурса.

Пусть X – некоторое частичное решение. Обозначим через G множество очередных операций, не вошедших в X. Множеству G поставим в соответствие множество переменных G, определяемое равенством

$$G = \left\{ x_{1j_1}\left(f'_{1j_1}\right),\ldots,x_{nj_n}\left(f'_{njn}\right)\right\}$$

где f'_{ij} – минимально возможное время окончания операции

Очередная операция (i, j) называется инцидентной с операцией (g, h), если при их выполнении используется один и тот же ресурс, причем $G'_{qg} \le G'_{ij} \le f'_{qh}$ запись $(i,j)\overset{k}{\longleftrightarrow}(q,h)$ означает, что операции инциденты по k-ому ресурсу. Определим G_k как подмножество множества G, для элементов которого выполняется

$$\forall(i,j),(q,h)\in G_k\left[(i,j)\overset{k}{\longleftrightarrow}(q,h)\right]$$

$$\sum_{(i,j)\in G_k} G^k_{i,j} > G^k$$

Множество G_k называется множеством операций, конфликтных по k-ому ресурсу. Если $G_k \neq \varnothing$, то ресурс k-ого вида называется дефицитным. Дефицит ресурса определяется равенством

$$\Delta_k = \sum_{(i,j)\in G_k} G_{i,j} - G^k$$

Множество $G_1 \cup G_2 \cup \ldots \cup G_m$ обозначим через G^*.

При работе алгоритма возникает необходимость в определении номера последней операции l-ого требования из числа записанных в частичное решение, а также времени окончания этой операции. Номер такой операции обозначим через $g_l(X)$, а момент ее окончания через $t_l(X)$.

Для нахождения $g_l(X)$ и $t_l(X)$ необходимо просматривать записи частичного решения в обратном порядке до тех пор, пока впервые не встретится запись с $I = l$.

В соответствии с основной идеей метода ветвей и границ, для частичных решений X ищется оценка снизу $\underline{Q}(X)$.

Оценка $\underline{Q}(X)$ вычисляется следующим образом.

Перенумеруем требования так, чтобы выполнялись неравенства

$$t_1 \le t_2 \le \ldots \le t_n$$

где $t_i = t_i\left(X\right) + \sum_{j=g_i(X)+1}^{n_i} \tau_{ij}$.

Величина t_i определяет минимально возможное время начала последней операции.

Пусть t_1^*, t_2^*, …, t_n^* – моменты окончания обслуживания соответствующих требований.

$$t_i^* = \max\{t_1, \theta_i\} + \tau_{ini}$$

где θ_i – момент высвобождения необходимых ресурсов. Тогда $\underline{Q}(X) = t_n^*$.

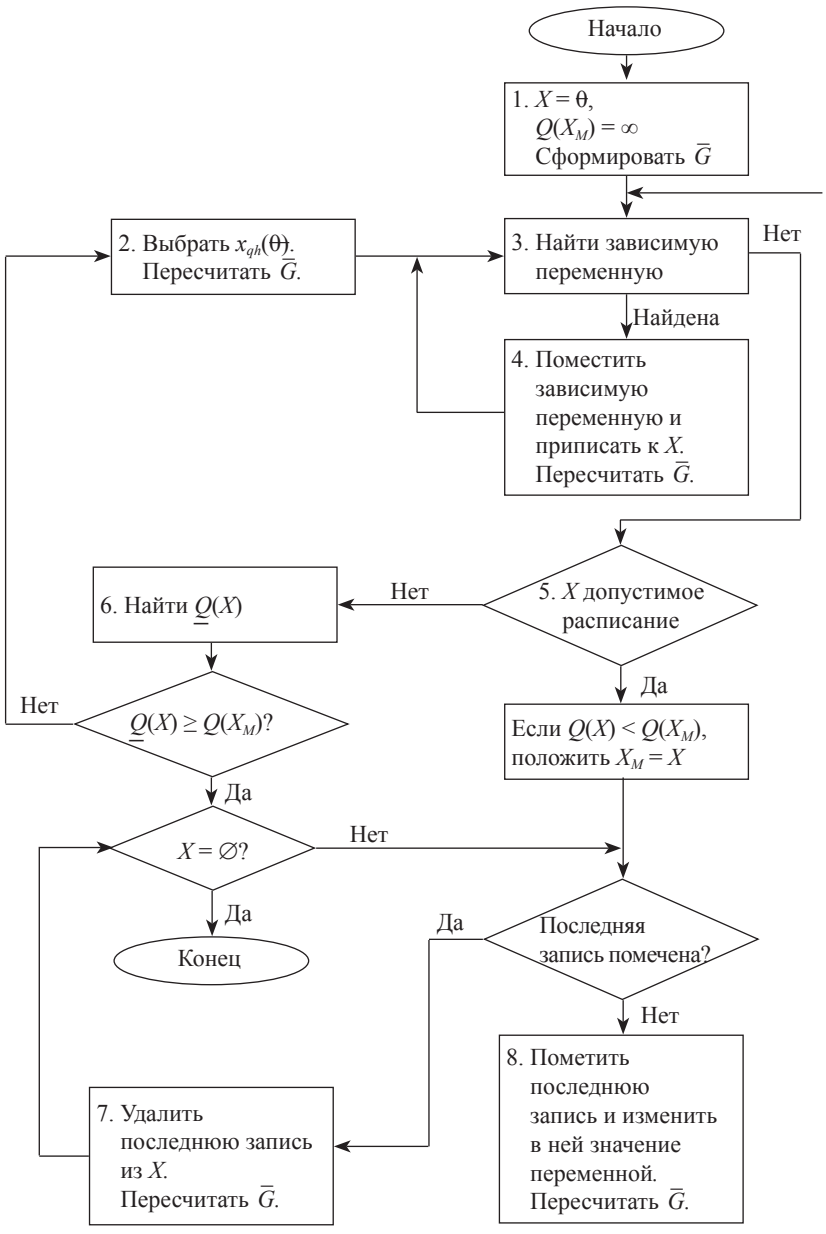

Рис. 1. Блок-схема алгоритма

Дадим теперь описание алгоритма. Его блок схема приведена на рис. 1. Ниже приводится описание блоков алгоритма.

Блок 1. $\overline{G} = \left\{x_{11}\left(f_{11}^{'}\right),...,x_{n1}\left(f_{n1}^{'}\right)\right\}$.

Блок 2. Выбирается ресурс с наибольшим дефицитом. Пусть это будет ресурс k-ого вида. Из подмножества $\overline{G}_k$ выбирается переменная $x_{qh}(\theta)$ доставляющая минимум вектору (t, i); переменная $x_{qh}(\theta)$ доставляет минимум вектору (t, i), если для любой другой переменной $x_{ij}(t) \in \overline{G}_k$, $0 < t$, либо $\theta = t$, но $q < i$. Смысл этого правила в том, что приоритет дается операции, выполнение которой позволяет наиболее быстро освободить дефицитный ресурс. Значение $x_{qh}(\theta)$ полагается равным единице. Вновь сформированная запись присоединяется к частичному решению. Выполнение блока заканчивается пересчетом подмножества $\overline{G}_k$. Переменная $x_{qh}(\theta)$ заменяется на $x_{qh+1}(t_q(x))$. У остальных переменных t заменяется на $t(X)$.

Блок 3. Ищется переменная $x_{ij}(t) \in \overline{G}$, доставляющая минимум вектору (t, i). Если выполняется

$$\forall_k[x_{qh}(\theta) \notin G_k]$$

то переменная $x_{qh}(\theta)$ считается зависимой и полагается равной единице. Смысл этой процедуры заключается в том, что ресурсы, необходимые для выполнения $x_{qh}(\theta)$ – операции, свободны и откладывание ее выполнения не имеет смысла.

Блок 4. Формируется очередная запись и приписывается к частичному решению X. Переменная $x_{qh}(\theta)$ заменяется в $\overline{G}$ на переменную $x_{qh+1}(t(X))$. Остальные переменные остаются без изменения.

Блок 5. X является допустимым решением, если $G = \varnothing$, т.е. все операции выполнены.

Блок 6. Для частичного решения X находится оценка $\underline{Q}(X)$.

Блок 7. Из частичного решения удаляется последняя запись q, h, θ, $x_{qh}(\theta)$. В множестве $\overline{G}$ переменная с первым индексом заменяется переменной $x_{qh}(\theta)$. Если значение $x_{qh}(\theta)$ было равным нулю, у всех переменных $x_{ij}(t)$ также, что $\exists$ к $x_{ij}(t) \overset{k}{\longleftrightarrow} x_{qh}(\theta)$, пересчитываются индексом t. Тем самым восстанавливается множество $\overline{G}$, соответствующее предыдущей вершине дерева вариантов.

Блок 8. Последняя запись частичного решения изменяется; значение переменной полагается равным нулю, сама запись метится. Восстанавливается прежнее значение множества $\overline{G}$. В множестве $\overline{G} \setminus x_{ij}(t)$, аналогично тому, как это делалось во втором блоке, выбирается переменная $x_{qh}(\theta)$. Формируется новая запись и присоединяется к частичному решению.

Вышеописанный алгоритм находит оптимальное расписание за конечное число шагов.

Доказательство этого утверждения носит стандартный характер и поэтому здесь не приводится.

Расписание с минимальным временем реализации (из числа найденных) обозначается через X_m. Каждое вновь найденное расписание X сравнивается с расписанием X_m, которое хранится в памяти вычислительной машины. Если $Q(X) < Q(X_m)$, то расписание X запоминается в качестве X_m.

По окончании работы алгоритма X_m – оптимальное расписание.

Литература

1. Чернявский А.Л. Алгоритм для решения комбинаторных задач, основанные на методе неявного перебора / Автоматика и телемеханика, №2, 1972.

2. Беркетов Г.А. К вопросу о решении обобщенной задачи построения расписания /Сб. Математические методы решения инженерных задач – М.: МО СССР, 1978.

3. Brooks G.H., White C.R. An algorithm for finding optimal or near optimal solutions to the production scheduling problem. J. Indust. Eng., V.16, №1, 1965.

4. Shrade L. Solving resource – constrained network problems by implicit enumeration nonpreemptive case. Oper. Res., V. 188, №2, 1970.

References

1. Chernjavski A.L. An algorithm for solving combinatorial problems based on the implicit enumeration / Avtomatika i telemehanika, №2, 1972.

2. Berketov A.G. To the question of the solution of a generalized problem for scheduling / Sb. Matematicheskie metody resheniya inzhenernyh zadach – M.: MO SSSR, 1978.

3. Brooks G.H., White C.R. An algorithm for finding optimal or near optimal solutions to the production scheduling problem. J. Indust. Eng., V. 16, №1, 1965.

4. Shrade L. Solving resource – constrained network problems by implicit enumeration nonpreemptive case. Oper. Res., V. 188, №2, 1970.