

ЛИНЕЙНАЯ АППРОКСИМАЦИОННАЯ СХЕМА ДЛЯ МНОГОПРОЦЕССОРНОЙ ЗАДАЧИ OPEN SHOP*)

Г. Д. Воегингер, С. В. Севастьянов

Для r -стадийной задачи open shop с идентичными параллельными процессорами на каждой стадии и критерием «минимум длины расписания» строится аппроксимационная схема временной сложности $O(nrm + C(m, \varepsilon))$, где n — число работ, m — общее число процессоров, а $C(m, \varepsilon)$ — функция, не зависящая от n .

Введение

Задачи теории расписаний — как, впрочем, большинство задач дискретной оптимизации — традиционно относятся к труднорешаемым задачам. Эффективные алгоритмы точного решения, как правило, удается построить лишь для задач небольших размерностей, а попытки распространить эти результаты на задачи бóльших размерностей наталкиваются на непреодолимую стену NP-трудности. Поэтому приходится отказываться от нахождения точных решений и ограничиваться приближенными. Но и в этом случае, как показывают современные результаты для ряда оптимизационных задач, мы сталкиваемся с некоторым «пределом приближаемости», который для разных задач различен. Для одних задач существует полиномиальная по времени аппроксимационная схема (PTAS), когда при любом фиксированном $\varepsilon > 0$ удается предложить полиномиальный алгоритм \mathcal{A}_ε с относительной погрешностью, не превосходящей $1 + \varepsilon$. (Временная сложность алгоритма \mathcal{A}_ε естественно зависит от ε , но при фиксированном ε полиномиальна от длины записи исходной информации.) Для других задач при некоторых константах C удается построить C -приближающие алгоритмы (с относительной погрешностью $f(S^\mathcal{A})/f(S^*) \leq C$, где f — целевая функция, S^* — оптимальное решение, а $S^\mathcal{A}$ — решение, полученное алгоритмом \mathcal{A}), но не

*) Работа второго автора выполнена при финансовой поддержке Российского фонда фундаментальных исследований (код проекта 96–01–01591).

существует аппроксимационной схемы. В этом случае существует «барьер приближаемости», т. е. такая константа C , что для любого $C' < C$ из существования C' -приближающего алгоритма вытекало бы соотношение $P = NP$. Наконец, для задач третьей категории ни при какой константе C невозможно построение полиномиального C -приближающего алгоритма (при условии $P \neq NP$). (Для таких задач возможна дальнейшая классификация по уровням их сложности, но мы не будем здесь касаться этих вопросов.)

Естественно, первый вопрос, который ставится при анализе сложности той или иной NP -трудной задачи, состоит в определении категории сложности, к которой относится данная задача. Этот же вопрос интересует нас применительно к задачам теории расписаний. Для большинства задач теории расписаний этот вопрос пока остается открытым. В то время как для многих задач уже построены C -приближающие алгоритмы, лишь для небольшого числа «расписательских» задач построены аппроксимационные схемы или доказано их несуществование. В области многостадийных задач теории расписаний одним из таких редких счастливых случаев является задача *open shop* с фиксированным числом машин m и критерием «минимум длины расписания» (или в стандартной записи [2] $Om||C_{\max}$). В [5] мы описали вариант PTAS для задачи $Om||C_{\max}$, который при фиксированных m и ε имеет временную сложность $O(n \log n)$. Вместе с тем в [6] было показано, что если m не ограничено константой, то для данной задачи не существует PTAS (при условии $P \neq NP$), поскольку уже константа $C = 5/4$ является для нее «барьером приближаемости». Тем самым в классификации задач по степени приближаемости мы провели четкую границу между случаями, когда m ограничено константой, и случаями, когда m является переменной. Однако все еще остается открытым вопрос о существовании *вполне полиномиальной* схемы (FPTAS) для задачи $Om||C_{\max}$, т. е. схемы, временная сложность которой полиномиально зависела бы от $1/\varepsilon$.

В то время как построенная в [5] полиномиальная схема для задачи $Om||C_{\max}$ дает ответ на один из теоретических вопросов теории сложности, она не дает по-настоящему эффективного средства решения практических задач со сколь угодно высокой точностью. Действительно, аддитивная константа, входящая в оценку сложности этой схемы, зависит от временной сложности построения оптимального расписания для так называемых больших работ. Количество последних оценивается сверху двойной экспонентой от $1/\varepsilon$, а именно величиной $m(m/\varepsilon)^{2^{m/\varepsilon}}$. Ясно, что даже для небольших значений m и $1/\varepsilon$ эта величина огромна. Таким образом, для $(1 + \varepsilon)$ -приближенного решения задач с числом работ $n \leq m(m/\varepsilon)^{2^{m/\varepsilon}}$ наша схема вырождается в один из алгоритмов их

точного решения. Ввиду NP-трудности задачи open shop такие алгоритмы имеют, по-видимому, не менее чем экспоненциальную от n временную сложность. (В качестве упражнения предлагаем читателю взять значения $m = 3$, $\varepsilon = 1/3$ и оценить, для каких значений n схема не дает эффективного средства даже для $4/3$ -приближенного решения задачи $O3||C_{\max}$. Мы сравниваем с этой конкретной задачей, поскольку для нее известен $4/3$ -приближающий алгоритм линейной от n временной сложности без огромных аддитивных констант [4].)

В настоящей работе предлагается другой вариант аппроксимационной схемы. За счет нового способа деления работ на большие, средние и малые удастся, во-первых, существенно уменьшить число больших работ (до «простой» экспоненты $3 \cdot 2^{m/\varepsilon}$). Во-вторых, использование «полужадной» схемы достройки расписания для средних и малых работ (вместо жадной схемы в PTAS из [5]) позволяет заменить $O(n \log n)$ на $O(n)$ и избавиться от сомножителя при n , зависящего от ε . Таким образом, от FPTAS нашу схему отделяет лишь аддитивная константа (в оценке трудоемкости схемы), экспоненциально зависящая от m/ε .

Вместо классической задачи $Om||C_{\max}$ мы применяем новую схему к более общей — многопроцессорной r -стадийной задаче $Or(Pm)||C_{\max}$, где на каждой из r стадий имеется ограниченное количество параллельных идентичных процессоров. По-прежнему считаем, что общее число процессоров m ограничено константой.

Оставшаяся часть статьи организована следующим образом. В разделе 1 приводится формальная постановка задачи и содержатся вспомогательные результаты. В разделе 2 описывается схема \mathcal{A}_{GC} жадной достройки произвольного частичного расписания до полного, требующая линейного от n числа действий (вместо n^2 действий, возникающих в прямой реализации жадного принципа достройки расписания). В разделе 3 приводится описание и анализ аппроксимационной схемы. При этом уточняется, как следует модифицировать схему жадной достройки, чтобы избавиться от коэффициента при n , зависящего от ε . Раздел 4 содержит размышления о возможных путях дальнейшего совершенствования схемы.

1. Формулировка задачи, обозначения, вспомогательные результаты

В многопроцессорной r -стадийной системе open shop имеется n работ $\{J_1, \dots, J_n\}$ и r машинных цехов $\mathcal{M}_1, \dots, \mathcal{M}_r$; i -й цех состоит из m_i идентичных машин (процессоров) $M_\nu \in \mathcal{M}_i$; $m = \sum m_i$ — общее число машин во всех цехах. Работа J_j , $1 \leq j \leq n$, состоит из r операций $\sigma_1^j, \dots, \sigma_r^j$. Операция σ_i^j может быть выполнена любой из машин

$M_\nu \in \mathcal{M}_i$ за p_i^j единиц времени. В каждый момент времени любая работа может выполняться не более чем на одной машине, и каждая машина может выполнять не более одной операции. Порядок выполнения операций каждой работы не задан заранее и должен быть выбран, причем порядок прохождения по цехам разных работ может быть выбран различным. Прерывания при выполнении любой операции не допускаются. *Расписанием*, задающим процесс выполнения операций $\mathcal{O} = \{\sigma_i^j \mid j = 1, \dots, n; i = 1, \dots, r\}$, называется пара отображений $M : \mathcal{O} \rightarrow \{M_1, \dots, M_m\}$ и $S : \mathcal{O} \rightarrow \mathbb{R}^+$, где $M(\sigma_i^j)$ назначает машину-исполнителя операции σ_i^j , а $S(\sigma_i^j) \doteq s_i^j$ задает момент начала выполнения операции. Расписание называется *допустимым*, если оно удовлетворяет перечисленным выше технологическим требованиям. Требуется найти допустимое расписание минимальной длины, т. е. расписание S с минимальным моментом окончания последней операции ($C_{\max}(S)$).

Пусть C_{\max}^* обозначает оптимум задачи, $L_i = \sum_{j=1}^n p_i^j$ — суммарную нагрузку i -го цеха, $\hat{L}_i = L_i/m_i$ — среднемашинную нагрузку i -го цеха, $\hat{L}_{\max} = \max_{i=1, \dots, r} \hat{L}_i$, $d_j = \sum_{i=1}^r p_i^j$ — продолжительность работы J_j , а $d_{\max} = \max_j d_j$. Ясно, что для задачи $O(P) \parallel C_{\max}$ величины \hat{L}_{\max} и d_{\max} являются нижними оценками оптимума, т. е.

$$C_{\max}^* \geq \max\{\hat{L}_{\max}, d_{\max}\}. \quad (1)$$

В [3] было показано, что любой жадный алгоритм \mathcal{A}_G (в том числе алгоритмы, достраивающие пустое расписание по схеме \mathcal{A}_{GC}) позволяет находить для задачи $O(P) \parallel C_{\max}$ *плотное* расписание S длины

$$C_{\max}(S) \leq \hat{L}_{\max} + d_{\max}. \quad (2)$$

Таким образом, из (1) и (2) следует, что алгоритм \mathcal{A}_G строит приближенное расписание для задачи $O(P) \parallel C_{\max}$ с верхней оценкой относительной точности, равной 2.

2. Схема \mathcal{A}_{GC} жадной достройки расписания многопроцессорной системы открытого типа

В отличие от схемы \mathcal{A}_{GR} построения плотного допустимого расписания, изложенной в [1], мы не будем тратить усилия на поддержание постоянных приоритетных порядков на множествах операций каждой работы и каждой машины, что позволит понизить временную сложность алгоритма. С другой стороны, построение расписания ведется в более сложных условиях, когда приходится учитывать наличие уже

имеющегося частичного расписания. (В описываемой ниже схеме \mathcal{A}_{GC} предполагается, что множество операций, для которых задано частичное расписание, может быть произвольным.) Это накладывает дополнительные ограничения на способы построения искомого расписания и требует дополнительных вычислительных затрат. Однако благодаря предложенной схеме удастся сохранить низкую временную сложность алгоритма. В следующем разделе данная схема применяется для достройки частичного расписания, первоначально заданного для некоторого подмножества так называемых больших работ.

Перейдем к детальному описанию работы алгоритма по схеме \mathcal{A}_{GC} . (Аббревиатура «GC» происходит от английского “greedily completing” — жадно достраивающий.)

Пусть задано частичное расписание для исходной системы, т. е. для некоторых операций $\{\sigma_i^j\}$ (будем называть их «старыми») уже заданы обслуживающие их процессоры $M(\sigma_i^j)$ и моменты $\{s_i^j\}$ начала их выполнения. Требуется назначить исполнителей и моменты начала для оставшихся («новых») операций (будем называть это «назначить операцию») без изменения уже имеющегося расписания так, чтобы длина полученного расписания была по возможности минимальной. Здравой идеей является как можно более полное заполнение промежутков простоя между старыми операциями, и мы предлагаем делать это следующим алгоритмом жадного типа.

Текущее (частичное) расписание будет храниться в двух матрицах размера $r \times n$: в матрице M будет задаваться процессор, обслуживающий операцию σ_i^j ($1 \leq i \leq r$, $1 \leq j \leq n$), а в матрице s — момент начала операции σ_i^j . (Если для операции σ_i^j расписание еще не назначено, то $M(\sigma_i^j) = \text{nul.}$) Кроме того, все уже назначенные в расписание операции будут храниться в виде «списка операций» \mathcal{L} . Предполагаем, что возможен прямой доступ к любому элементу списка \mathcal{L} , если известен его абсолютный адрес. Кроме того, при каждом элементе из \mathcal{L} будут храниться ссылки на адреса других элементов списка. В частности, для каждой операции σ_i^j в списке \mathcal{L} хранится номер работы j , которой принадлежит эта операция, номер обслуживающего процессора $M(\sigma_i^j)$, тип операции (старая или новая), момент ее окончания $c(\sigma_i^j)$, а также адреса предыдущей и последующей операций работы J_j в списке \mathcal{L} , адреса предыдущей и последующей операций машины $M(\sigma_i^j)$ в списке \mathcal{L} и адреса предыдущего и последующего элементов списка \mathcal{L} по ключу $c(\sigma_i^j)$.

В процессе работы алгоритма один раз просматривается весь список \mathcal{L} по ключу c , в подходящие моменты в расписание назначаются какие-то новые операции и добавляются в список \mathcal{L} . Текущую просматриваемую операцию списка \mathcal{L} называем *актуальной* операцией.

Процесс просмотра сопровождается скачкообразным неубывающим изменением значения переменной τ , которую называем *текущим моментом*. Значение τ всегда совпадает с моментом окончания актуальной операции. *Текущей операцией* работы J_j (машины M_i) называем ближайшую (по ключу c) еще не просмотренную операцию работы J_j (машины M_i) из списка \mathcal{L} ; при этом не имеет значения, выполняется ли операция в момент τ или еще не начата. Ближайшую (по ключу c) непросмотренную старую/новую операцию из списка \mathcal{L} будем коротко называть *ближайшей старой/новой операцией*. В ходе работы алгоритма хранятся адреса обеих ближайших операций, а также поддерживаются массивы $\Phi[1 \dots n]$ и $\hat{j}[1 \dots m]$ адресов текущих операций каждой работы J_j ($j = 1, \dots, n$) и каждой машины M_i ($i = 1, \dots, m$).

Информация о новых, уже назначенных в расписание (и включенных в список \mathcal{L}), но еще не просмотренных операциях (будем называть их «полуопределенными») хранится в виде «кучи» \mathcal{H} , полупорядоченной по ключу c : в корневой вершине этой кучи всегда находится операция с наименьшим значением ключа и по каждой цепочке, идущей из корня в висячую вершину, значение ключа не убывает. При каждой вершине кучи хранится пара: адрес операции в списке \mathcal{L} и значение ключа. «Полуопределенность» операции в списке \mathcal{L} означает, что у нее определены все параметры, кроме двух: ссылки на предыдущую и ссылки на следующую операции списка \mathcal{L} по ключу c . Эти ссылки становятся известными после «просмотра» операции, т. е. после назначения ее актуальной операцией.

В каждый момент времени все новые неназначенные операции хранятся в трех списках: в списке A_i операций, *разрешенных* для назначения на машину M_i ($i = 1, \dots, m$), а также в списках R_j^J ($j = 1, \dots, n$) и R_i^M ($i = 1, \dots, m$) операций работы J_j (машины M_i), временно запрещенных для назначения их в расписание. Операция o' попадает в список запрещенных после ее удаления из какого-либо списка A_i по причине неизбежного перекрытия o' с текущей операцией работы J_j (текущей старой операцией машины M_i) в случае назначения o' в текущий момент. Для каждой операции o' , перемещенной из списка A_i в какой-то список R_j^J , запоминается машина $M(o')$, из списка которой она была взята. (Для операций, помещенных в список R_i^M , этого не требуется, так как все операции в R_i^M взяты из A_i .) Поскольку алгоритм работает по жадному принципу, то простой машины M_i в течение какого-то интервала времени возможен лишь при пустом списке A_i . (Отсюда следует, что если в какой-то момент τ список A_i не пуст, то машина M_i не простаивает.)

Алгоритм состоит из нулевого шага (или шага *инициализации*) и цикла пошагового просмотра элементов списка \mathcal{L} по неубыванию ключа c .

На шаге **инициализация** задаются начальные значения списков и переменных. Полагается $\tau := 0$. Список \mathcal{L} состоит только из старых операций. При этом в \mathcal{L} добавляется m «старых» операций $\bar{o}_1, \dots, \bar{o}_m$, относящихся к фиктивным работам I_{n+1}, \dots, I_{n+m} , выполняемых соответственно на машинах M_1, \dots, M_m и имеющих моменты окончания $s(\bar{o}_i) = 0$. Все новые операции каждого цеха \mathcal{M}_ν ($\nu = 1, \dots, r$) находятся в списке \mathcal{O}_ν . Для каждой машины $M_i \in \mathcal{M}_\nu$ полагается $A_i := \mathcal{O}_\nu$. (Таким образом, каждая операция $o_\nu^j \in \mathcal{O}_\nu$ представлена в m_ν независимых списках A_i .) Списки R_j^J ($j = 1, \dots, n$), R_i^M ($i = 1, \dots, m$) и \mathcal{H} сначала пусты. Находятся адреса текущих операций всех работ (массив $\Phi[1 \dots n]$) и всех машин (массив $\hat{j}[1 \dots m]$), а также адрес ближайшей старой операции. Для получения всей этой информации достаточно одного просмотра списка \mathcal{L} по ключу s .

Цикл просмотра списка \mathcal{L} состоит из шагов, на каждом из которых просматривается только одна (актуальная) операция из этого списка. Каждый шаг начинается с определения новой актуальной операции. Она выбирается из двух операций: ближайшей старой и ближайшей новой, адрес которой хранится в корневой вершине кучи \mathcal{H} . (На первом шаге цикла, когда куча \mathcal{H} еще пуста, выбор ограничивается единственной ближайшей старой операцией.) Одновременно изменяется значение τ . Если выбрана старая операция, то она сразу же перестает быть «ближайшей старой», а в качестве таковой берется следующая (по ключу s) операция из списка \mathcal{L} . Если же выбрана новая операция, то сначала доопределяются ее ссылки на предыдущую и последующие операции из списка \mathcal{L} (в качестве таковых берутся предыдущая актуальная и ближайшая старая операции и заодно корректируются ссылки у этих двух операций), а затем операция удаляется из корня кучи \mathcal{H} и куча перестраивается (что может быть сделано за время $O(\log n_{\mathcal{H}})$, где $n_{\mathcal{H}}$ — максимально возможная мощность кучи \mathcal{H}), одновременно определяется новое значение для «ближайшей новой» операции. Кроме того, актуальная операция перестает быть текущей операцией как своей работы, так и своей машины, и эти титулы переходят соответственно к следующей (в списке \mathcal{L}) операции этой работы и следующей операции этой машины.

На каждом шаге также принимается решение о назначении в расписание в момент τ каких-то (не более двух) новых операций. Это назначение производят две процедуры, которые называются «назначение работы» и «загрузка машины». Пусть актуальная операция $o_k^{j'}$ выполняется на машине $M_{i'}$.

Назначение работы предназначено для постановки в расписание в момент τ одной из еще не назначенных операций работы $J_{j'}$. Поиск такой операции производится в списке $R_{j'}^J$. (Использовать для этой

цели списки A_i не имеет смысла, так как если список A_i не пуст, то согласно сделанному выше замечанию машина M_i не простаивает. Следовательно, никакая операция не может быть назначена на эту машину в текущий момент.) Последовательно просматриваются операции из списка $R_{j'}^J$, и для каждой операции o' выполняются следующие действия.

- Если o' перекрывается с текущей операцией машины $M_i = M(o')$ и эта текущая операция — новая, то o' перемещается в список A_i .
- Если o' перекрывается с текущей операцией машины $M_i = M(o')$ и эта операция — старая, то o' перемещается в список R_i^M .
- Если o' не перекрывается с текущей операцией машины $M(o')$, но перекрывается с текущей операцией работы $J_{j'}$, то o' остается в списке $R_{j'}^J$.
- Наконец, если o' не перекрывается ни с текущей операцией машины $M(o')$, ни с текущей операцией работы $J_{j'}$, то операция o' назначается в расписание в момент τ и одновременно выполняются следующие действия:
 - * o' удаляется из списка $R_{j'}^J$;
 - * o' назначается текущей операцией работы $J_{j'}$ и машины $M(o')$ и определяются соответствующие ссылки на предыдущие и последующие операции по обеим цепочкам в списке \mathcal{L} ;
 - * o' помещается в список \mathcal{L} и в кучу \mathcal{H} и за время $O(\log n_{\mathcal{H}})$ выполняется необходимая перестройка кучи.

Процедура заканчивается, как только какая-то операция из списка $R_{j'}^J$ назначается в расписание либо когда все операции из списка $R_{j'}^J$ просмотрены. Ясно, что во время текущего шага будет назначено не более одной операции из списка $R_{j'}^J$.

Загрузка машины предназначена для постановки в расписание в момент τ одной из еще не назначенных операций машины $M_{i'}$. Вначале проверяется, является ли актуальная операция старой, и если «да», то всё содержимое списка $R_{i'}^M$ перемещается в список $A_{i'}$. Затем (независимо от типа актуальной операции: старая она или новая) последовательно просматривается список $A_{i'}$ и для каждой операции $\sigma_k^j \in A_{i'}$ выполняются следующие действия.

- Если σ_k^j уже назначена в расписание (такое возможно в силу относительной независимости списков $\{A_i\}$), то σ_k^j удаляется из списка $A_{i'}$.
- Если σ_k^j перекрывается с текущей операцией машины $M_{i'}$ (очевидно, текущая операция является «старой»), то σ_k^j перемещается из $A_{i'}$ в $R_{i'}^M$.

- Если σ_k^j перекрывается с текущей операцией работы J_j , то σ_k^j перемещается из A_i в R_j^J .
- Если σ_k^j не перекрывается с обеими операциями, то она назначается в расписание в момент τ . При этом, как и в процедуре «назначение работы», выполняются все действия, положенные для только что назначенной операции, и просмотр списка A_i прекращается.

Описание алгоритма \mathcal{A}_{GC} завершено.

Лемма 1. Пусть в m -процессорной r -стадийной системе open shop задано частичное расписание S , в котором для v_k операций цеха \mathcal{M}_k ($k = 1, \dots, r$) расписание полностью определено, а для n_k (новых) операций цеха \mathcal{M}_k расписание предстоит доопределить. Пусть также информация о частичном расписании S задана в том виде, как это требуется в алгоритме \mathcal{A}_{GC} . Тогда алгоритм \mathcal{A}_{GC} достраивает исходное частичное расписание для новых операций за время $O\left(\sum_{k=1}^r n_k(rm_k + v_k + \log n_{\mathcal{X}})\right)$, где $n_{\mathcal{X}}$ — максимально возможное число одновременно выполняемых новых операций.

Доказательство. Работа алгоритма на каждом шаге состоит из нахождения новой актуальной операции и выполнения процедур «назначение работы» и «загрузка машины». Если в качестве актуальной операции выбрана ближайшая старая, то это требует лишь $O(1)$ действий, тогда как при выборе ближайшей новой операции в качестве актуальной на перестройку кучи \mathcal{H} нужно потратить $O(\log n_{\mathcal{X}})$ действий. Следовательно, суммарное по всем шагам время, затрачиваемое на выбор актуальной операции, не превосходит $O\left(\sum_{k=1}^r (v_k + n_k \log n_{\mathcal{X}})\right)$.

Суммарная сложность вычислительных операций, выполняемых внутри процедур, складывается из временной сложности действий, связанных с перемещениями новых операций между списками A_i , R_j^J и R_i^M , и сложности действий, связанных с назначением новых операций в расписание. Первая сложность пропорциональна количеству перемещений. При этом для каждой машины $M_i \in \mathcal{M}_k$ каждая из n_k новых операций, первоначально входящих в список A_i , не более $r - 1$ раз может быть удалена из списка A_i (в соответствующий список R_j^J) по причине ее перекрытия с другой операцией той же работы и не более $r - 1$ раз возвращена в список A_i , а также не более s_i раз удалена (возвращена) по причине ее перекрытия с одной из старых операций на машине M_i , где s_i — количество старых операций на машине M_i . Последнее удаление происходит при назначении операции в расписание. Таким образом,

общее число перемещений каждой новой операции из списка A_i равно $O(r + s_i)$, а суммарное по всем машинам число перемещений новых операций не превосходит

$$O\left(\sum_{k=1}^r \sum_{M_i \in \mathcal{M}_k} n_k(r + s_i)\right) = O\left(\sum_{k=1}^r n_k(rm_k + v_k)\right).$$

Наконец, при назначении новой операции в расписание наибольшую временную сложность имеет процедура добавления операции в кучу \mathcal{H} с последующей перестройкой кучи, что требует $O\left(\sum_{k=1}^r n_k \log n_{\mathcal{H}}\right)$ действий. Как легко убедиться, сложение полученных выше оценок для числа вычислительных операций дает требуемую в лемме оценку сложности алгоритма. Лемма 1 доказана.

3. Полиномиальная аппроксимационная схема для многопроцессорной задачи open shop с фиксированным числом машин

В этом разделе будет рассмотрена многопроцессорная задачи open shop, в которой число работ n является переменной, число стадий r фиксировано, а число машин на стадии i , $1 \leq i \leq r$, ограничено константой m_i . Будет показано, что для этой задачи существует полиномиальная аппроксимационная схема линейной временной сложности от n .

Предлагаемая аппроксимационная схема для $Or(Pm) \parallel C_{\max}$ использует идею «полужадного» алгоритма, строящего «полуплотные» расписания. В отличие от плотного расписания, в котором простые машины возможны лишь в те моменты, когда нет готовых к выполнению на ней операций, в полуплотном расписании будет допускаться некоторое относительно небольшое количество «насильственных» простоев.

Другой решающей идеей является разбиение всего множества работ на три подмножества: «больших», «средних» и «малых» работ. Для двух вещественных чисел $\alpha' > \alpha'' > 0$ определяем три подмножества работ: $L = \{J_j \in \mathcal{J} \mid d_j \geq \alpha' \hat{L}_{\max}\}$, $M = \{J_j \in \mathcal{J} \mid \alpha'' \hat{L}_{\max} \leq d_j < \alpha' \hat{L}_{\max}\}$ и $S = \{J_j \in \mathcal{J} \mid d_j < \alpha'' \hat{L}_{\max}\}$, которые будем называть соответственно *большими*, *средними* и *малыми*. Операции больших, средних и малых работ также будем называть *большими*, *средними* и *малыми* (независимо от их действительных размеров). Выбранные числа α', α'' должны удовлетворять следующим требованиям:

(а) число $|L|$ больших работ должно быть ограничено константой (для заданного ε);

- (b) суммарная длина средних работ должна быть не больше $\varepsilon \hat{L}_{\max}$;
 (c) отношение α''/α' должно быть достаточно малым, чтобы обеспечить неравенство

$$\alpha' + \alpha'' + \alpha''|L| \leq \varepsilon. \quad (3)$$

Схема представляет собой семейство алгоритмов \mathcal{A}_ε , обеспечивающих полиномиальное построение $(1 + \varepsilon)$ -приближенного расписания при любом фиксированном $\varepsilon > 0$. Опишем в общем виде работу произвольного алгоритма \mathcal{A}_ε .

Алгоритм \mathcal{A}_ε .

Искомое расписание строится за 4 шага.

Шаг 1. Если $\varepsilon \geq 1$ или $d_{\max} \leq \varepsilon \hat{L}_{\max}$, то искомое расписание S строится с помощью жадного алгоритма \mathcal{A}_G . В противном случае выполняется шаг 2.

Шаг 2. Находится разбиение множества работ на три подмножества: L , M и S (больших, средних и малых работ), удовлетворяющее требованиям (a)–(c).

Шаг 3. Для работ из множества L находится оптимальное расписание S^L .

Шаг 4. С помощью полужадной модификации схемы \mathcal{A}_{GC} расписание S^L достраивается для остальных работ (из множеств M и S).

Конец.

Теорема 1. Для любого действительного числа $\varepsilon > 0$, целых r , m и любых входных данных задачи $Or(Pm) \parallel C_{\max}$ с n работами алгоритм \mathcal{A}_ε строит расписание S длины

$$C_{\max}(S) \leq (1 + \varepsilon) C_{\max}^*. \quad (4)$$

Временная сложность алгоритма равна $O(n)$, причем мультипликативная константа является полиномом от m и не зависит от ε .

Доказательство. Если $\varepsilon \geq 1$ или $d_{\max} \leq \varepsilon \hat{L}_{\max}$, то согласно (2) требуемое расписание S строится на первом шаге с помощью алгоритма \mathcal{A}_G . Пусть $\varepsilon \in (0, 1)$, $d_{\max} > \varepsilon \hat{L}_{\max}$ и пусть на втором шаге алгоритма \mathcal{A}_ε найдены числа α' , α'' , удовлетворяющие требованиям (a)–(c). (Чуть позже мы опишем этот шаг более подробно.) Шаг 3 не нуждается в детальном описании. Ясно, что построение оптимального расписания S^L для работ из множества L может быть осуществлено каким-либо алгоритмом переборного типа. Временная сложность этого алгоритма войдет в оценку сложности всего алгоритма в качестве слагаемого, величина которого не зависит от n , но зависит от числа больших работ и числа машин. Поскольку две последние величины ограничены константой, то указанное слагаемое становится аддитивной константой.

Для достройки расписания для средних и малых работ на шаге 4 мы могли бы применить схему \mathcal{A}_{GC} . Тогда согласно лемме 1 временная сложность шага 4 оценивалась бы величиной $O\left(\sum_{k=1}^r n_k(rm_k + v_k + \log n_{\mathcal{A}})\right) \leq O((rnm + rn|L|))$, а значит, мультипликативная константа при n зависела бы от ε . Покажем, как избавиться от этой зависимости, слегка пожертвовав свойством плотности расписания.

Произведем следующее небольшое изменение в работе описанной выше процедуры «загрузка машины». Теперь до начала просмотра списка A_i будет оцениваться длина оставшегося интервала простоя на текущей машине, т. е. интервала от момента τ до начала выполнения текущей (старой) операции на данной машине; если эта длина меньше величины $\alpha'' \hat{L}_{\max}$, то процедура заканчивается и осуществляется переход к следующему шагу (независимо от того, что какие-то разрешенные операции из списка A_i вполне уместятся в данном интервале простоя и могли бы быть назначены в момент τ). Это гарантирует то, что никакая операция малой работы никогда не будет перемещена из списка A_i в список R_i^M . Значит, для каждой малой операции из цеха i суммарное количество ее перемещений не будет превосходить $O(rm_i)$ (напомним, что каждая операция представлена в m_i независимых списках A_i). Поэтому суммарное число перемещений по всем операциям малой работы есть $O(rm)$, а по всем малым работам — $O(nrm)$. Для средних операций все остается по-старому, поэтому суммарное количество их перемещений не превосходит $O(r|M|(m + |L|))$. Поскольку и количество больших, и количество средних работ ограничены константами (зависящими от ε), то число их перемещений дает вклад в общую оценку сложности в виде аддитивной константы. Таким образом, сложность шага 4 в новой версии не превосходит $O(nrm)$.

Покажем, что относительная погрешность описанного алгоритма действительно не превосходит ε . Пусть S^A обозначает полученное расписание, C_{\max}^A — его длину, а C_{\max}^L — длину расписания S^L . Очевидно, что $C_{\max}^* \geq C_{\max}^L$. Без ограничения общности можем предполагать, что *критической машиной*, которая заканчивает работу в момент C_{\max}^A , является машина $M_1 \in \mathcal{M}_1$.

Если последней операцией на машине M_1 является большая операция, то $C_{\max}^A = C_{\max}^L \leq C_{\max}^*$ и расписание S^A оптимально. Далее предполагаем, что в расписании S^A последняя операция на машине M_1 принадлежит работе J_k , которая не является большой. Отсюда следует, что $d_k \leq \alpha' \hat{L}_{\max}$. Пусть t — момент начала операции o_1^k , t_i^* — момент завершения последней большой операции в расписании S^L на машине

$M_i \in \mathcal{M}_1$ и $t^* = \max_{M_i \in \mathcal{M}_1} t_i^*$. Ясно, что

$$t^* \leq C_{\max}^* \leq C_{\max}^A = t + p_1^k$$

и никакая большая операция не выполняется в интервале $[t^*, C_{\max}^A]$ на какой-либо машине $M_i \in \mathcal{M}_1$. Сначала предположим, что никакая малая операция (даже частично) не выполняется на машине M_1 в интервале $[t^*, C_{\max}^A]$ (следовательно, J_k — средняя работа). Поскольку все простои машины M_1 в интервале $[t^*, C_{\max}^A]$ обусловлены выполнением операций работы J_k в других цехах, то суммарный простой на M_1 в интервале $[t^*, C_{\max}^A]$ не превосходит $d_k - p_1^k$, а суммарная нагрузка машины M_1 в этом же интервале не превосходит $p_1^k + \sum_{J_j \in M \setminus \{J_k\}} d_j$. Таким образом, длина

интервала $[t^*, C_{\max}^A]$ не превосходит суммарной длины средних работ, которая ввиду требования (b) не превосходит $\varepsilon \hat{L}_{\max}$. Отсюда следует, что

$$C_{\max}^A \leq t^* + \varepsilon \hat{L}_{\max} \leq (1 + \varepsilon) C_{\max}^*.$$

Теперь рассмотрим случай, когда существует малая операция σ_1^j , которая завершается на машине M_1 в момент $t' > t^*$. Простой на машине $M_i \in \mathcal{M}_1$ в интервалах $[0, t_i^*]$ и $[t_i^*, t]$ будем называть *внутренним* и *внешним* соответственно. Оценим суммарную величину внутреннего и внешнего простоя (I_1 и I_2 соответственно) по всем машинам $M_i \in \mathcal{M}_1$. Прежде всего заметим, что каждый интервал внутреннего простоя либо покрывается интервалом выполнения другой операции работы J_j (суммарная длина таких интервалов по всем машинам $M_i \in \mathcal{M}_1$ не превосходит $m_1 d_j \leq m_1 \alpha'' \hat{L}_{\max}$), либо является тем самым «насильственным» интервалом простоя, который предшествует началу выполнения какой-то большой операции на данной машине и не превосходит величины $\alpha'' \hat{L}_{\max}$. Суммарная длина этих интервалов не превосходит $|L| \alpha'' \hat{L}_{\max}$. Таким образом, $I_1 < (m_1 + |L|) \alpha'' \hat{L}_{\max}$.

Внешние простои обусловлены только выполнением операций работы J_k в других цехах. Поэтому суммарный по всем машинам $M_i \in \mathcal{M}_1$ внешний простой не превосходит $I_2 \leq m_1 (d_k - p_1^k)$. Так как каждая машина $M_i \in \mathcal{M}_1$ в любой момент времени из интервала $[0, t]$ либо занята, либо простаивает, а в интервале $[0, t]$ все машины цеха \mathcal{M}_1 могут быть заняты не более L_1 единиц времени, то

$$tm_1 \leq L_1 + I_1 + I_2 < L_1 + (m_1 + |L|) \alpha'' \hat{L}_{\max} + m_1 (d_k - p_1^k).$$

Отсюда следует, что

$$\begin{aligned} C_{\max}^A &= t + p_1^k < L_1/m_1 + d_k + (1 + |L|/m_1) \alpha'' \hat{L}_{\max} \\ &\leq (1 + \alpha' + \alpha'' + \alpha'' |L|) \hat{L}_{\max} \leq (1 + \varepsilon) C_{\max}^*. \end{aligned}$$

Таким образом, алгоритм \mathcal{A}_ε всегда гарантирует ε -оптимальность получаемых решений.

Опишем алгоритм нахождения чисел α', α'' , которые гарантируют разбиение всего множества работ на большие, средние и малые с выполнением требований (а)–(с).

Вычислим значения величин $\{d_j \mid j = 1, \dots, n\}$, $d_{\max} = \max d_j$ и \hat{L}_{\max} . Напомним, что мы предполагаем выполненными неравенства $d_{\max} > \varepsilon \hat{L}_{\max}$ и $\varepsilon < 1$. Можем считать, что $\varepsilon = m/N$ для некоторого целого $N > m$. Сначала будет представлен алгоритм временной сложности $O(n \log n)$, а затем будет показано, как получить тот же результат за время $O(n)$.

- Все работы нумеруются по невозрастанию d_j .
- Полагается $\alpha_1 = \frac{\varepsilon}{2}$, $\alpha_0 = \infty$, $E_1 = [\alpha_1 \hat{L}_{\max}, \alpha_0 \hat{L}_{\max})$. Просматривается список работ, пока не выполнится условие $d_j < \alpha_1 \hat{L}_{\max}$; вычисляется $D_1 \doteq \sum_{d_j \in E_1} d_j$ и полагается $n'_1 = \frac{D_1}{\alpha_1 \hat{L}_{\max}} - 1$.
- Для $k = 2, 3, \dots$ выполняются следующие действия:

* находится α_k из равенства

$$\alpha_{k-1} + \alpha_k(1 + n'_{k-1}) = \varepsilon; \quad (5)$$

* определяется $E_k = [\alpha_k \hat{L}_{\max}, \alpha_{k-1} \hat{L}_{\max})$; продолжается просмотр списка работ, пока не выполнится неравенство $d_j < \alpha_k \hat{L}_{\max}$; находится $D_k \doteq \sum_{d_j \in E_k} d_j$ и полагается

$$n'_k = n'_{k-1} + \frac{D_k}{\alpha_k \hat{L}_{\max}}; \quad (6)$$

* если $D_k \leq \varepsilon \hat{L}_{\max}$, то $\{\alpha'' := \alpha_k; \alpha' := \alpha_{k-1}; \text{stop}\}$.

- Конец цикла по k .

Сначала убедимся, что при любом $i \geq 2$ выполнено неравенство $\alpha_i < \alpha_{i-1}$ (т. е. величины $\{\alpha_i\}$ определяют непустые и **непересекающиеся** интервалы $\{E_i\}$). Предположим противное, т. е. для некоторого $i \geq 2$ оказалось $\alpha_i \geq \alpha_{i-1}$. Тогда при $i = 2$ получаем $\varepsilon = \alpha_1 + \alpha_2(1 + n'_1) \geq \alpha_1 + \frac{\alpha_1 D_1}{\alpha_1 \hat{L}_{\max}} \geq \alpha_1 + \frac{d_{\max}}{\hat{L}_{\max}} \geq \frac{3}{2}\varepsilon$. Противоречие. При $i \geq 3$ получаем $\varepsilon = \alpha_{i-1} + \alpha_i(1 + n'_{i-1}) \geq \alpha_{i-1} \left(2 + n'_{i-2} + \frac{D_{i-1}}{\alpha_{i-1} \hat{L}_{\max}}\right) > \alpha_{i-1}(2 + n'_{i-2}) + \varepsilon > \varepsilon$. Противоречие.

Положим $n_i = |\{J_j \mid d_j \geq \alpha_i \hat{L}_{\max}\}|$. Индукцией по i докажем неравенство $n_i \leq n'_i$.

Так как в сумму D_1 в качестве слагаемого входит длина работы J_{j^*} ($d_{j^*} = d_{\max} > \varepsilon \hat{L}_{\max}$), а длины остальных $n_1 - 1$ работ, представленных в сумме D_1 , оцениваются снизу величинами $d_j \geq \frac{\varepsilon}{2} \hat{L}_{\max}$, то

$$D_1 > \varepsilon \hat{L}_{\max} + (n_1 - 1) \frac{\varepsilon}{2} \hat{L}_{\max} = (n_1 + 1) \frac{\varepsilon}{2} \hat{L}_{\max} = (n_1 + 1) \alpha_1 \hat{L}_{\max}.$$

Следовательно, $n_1 \leq n'_1$.

Пусть неравенство $n_{i-1} \leq n'_{i-1}$ установлено. Тогда

$$n_i \leq n_{i-1} + \frac{D_i}{\alpha_i \hat{L}_{\max}} \leq n'_{i-1} + \frac{D_i}{\alpha_i \hat{L}_{\max}} = n'_i,$$

что и требовалось доказать.

Далее убедимся, что оператор **stop** сработал и величины α' , α'' определены. Из $D_1 \geq d_{\max} \geq \varepsilon \hat{L}_{\max}$ и соотношений

$$\sum_{k=1}^N D_k \leq \sum_{j=1}^n d_j = \sum_{i=1}^r L_i \leq \hat{L}_{\max} \sum_i m_i = m \hat{L}_{\max} = N \varepsilon \hat{L}_{\max}$$

следует, что для одной из величин D_2, \dots, D_k выполнено неравенство $D_k \leq \varepsilon \hat{L}_{\max}$, которое является условием срабатывания оператора **stop**.

Наконец, убедимся, что величины α' , α'' , определенные согласно описанной выше процедуре, удовлетворяют требованиям (а)–(с). Требование (b) является условием срабатывания **stop** и, как уже отмечалось, выполнено. Требование (с) вытекает из соотношений

$$\alpha_{k-1} + \alpha_k(1 + n_{k-1}) \leq \alpha_{k-1} + \alpha_k(1 + n'_{k-1}) = \varepsilon.$$

Требование (а) будет непосредственно вытекать из верхней оценки на n_{k-1} , которую мы сейчас получим.

Для удобства выкладок примем $\varepsilon \hat{L}_{\max}$ за единицу и обозначим $\beta_i = \alpha_i / \varepsilon$. Тогда для $i = 3, \dots, k$ из (5) и (6) имеем

$$1 - \beta_{i-1} = \beta_i(1 + n'_{i-1}) = \beta_i(1 + n'_{i-2} + D_{i-1}/\beta_{i-1}) = \beta_i(1 - \beta_{i-2} + D_{i-1})/\beta_{i-1},$$

$$\text{или } \beta_i(1 - \beta_{i-2} + D_{i-1}) = \beta_{i-1}(1 - \beta_{i-1}).$$

При $i = 2$ получаем аналогичное соотношение $1 - \beta_1 = \beta_2(1 + n'_1) = \beta_2 D_1 / \beta_1$, или $\beta_2(1 - \beta_0 + D_1) = \beta_1(1 - \beta_1)$, где $\beta_0 = 1$. Таким образом, величины β_i находятся из рекуррентных соотношений

$$\beta_0 = 1; \quad \beta_1 = 1/2; \tag{7}$$

$$\beta_i(1 - \beta_{i-2} + D_{i-1}) = \beta_{i-1}(1 - \beta_{i-1}), \quad i = 2, \dots, k, \tag{8}$$

как функции от параметров $\{D_i\}$, удовлетворяющих соотношениям

$$D_i \geq 1; \quad \sum_{i=1}^{k-1} D_i \leq N.$$

Поскольку число n_{k-1} больших работ удовлетворяет неравенству

$$n_{k-1} \leq n'_{k-1} = \frac{1 - \beta_{k-1}}{\beta_k} - 1 \leq \frac{1}{\beta_k} - 2,$$

то при получении верхней оценки для n_{k-1} достаточно оценить снизу величину β_k при $k \in [2, N]$. Так как величина D_{k-1} присутствует лишь в последнем рекуррентном соотношении из (8), т. е.

$$\beta_k(1 - \beta_{k-2} + D_{k-1}) = \beta_{k-1}(1 - \beta_{k-1}),$$

то β_k принимает минимальное значение при максимально возможном D_{k-1} . Таким образом, мы можем считать выполненным равенство $\sum_{i=1}^{k-1} D_i = N$. Для каждого $k \in \{2, \dots, N\}$ найдем нижнюю оценку для величины β_k при соотношениях (7), (8) и

$$D_i \geq 1; \quad \sum_{i=1}^{k-1} D_i = N. \quad (9)$$

Вначале покажем, что при любом $i = 0, 1, 2, \dots$ величина β_i удовлетворяет неравенству

$$\beta_i \leq 1/2^i. \quad (10)$$

Для β_0 и β_1 неравенство выполнено. Пусть оно выполнено для $i \leq k$, где $k \geq 1$. Из (8) и $D_i \geq 1$ получаем

$$\beta_{k+1} = \frac{\beta_k(1 - \beta_k)}{1 - \beta_{k-1} + D_k} \leq \frac{\beta_k(1 - \beta_k)}{2 - 1/2^{k-1}}.$$

Выражение $\beta_k(1 - \beta_k)$ как функция от β_k возрастает на отрезке $[0, 1/2]$ и с учетом неравенства $\beta_k \leq 1/2^k$ принимает максимальное значение при $\beta_k = 1/2^k$. Поэтому

$$\beta_{k+1} \leq \frac{\frac{1}{2^k} \left(1 - \frac{1}{2^k}\right)}{2(1 - 1/2^k)} = \frac{1}{2^{k+1}},$$

т. е. (10) выполнено при $i = k + 1$.

Последовательно воспользовавшись рекуррентной формулой (8), получаем

$$\begin{aligned}\beta_k &= \frac{(1 - \beta_{k-1})(1 - \beta_{k-2}) \cdots (1 - \beta_1)\beta_1}{(1 - \beta_{k-2} + D_{k-1})(1 - \beta_{k-3} + D_{k-2}) \cdots (1 - \beta_1 + D_2)D_1} \\ &= \left(\text{ибо } \beta_1 = \frac{1}{2} \right) = \frac{1 - \beta_{k-1}}{2D_1} \prod_{i=1}^{k-2} \frac{1 - \beta_i}{1 - \beta_i + D_{i+1}} \geq (\text{см. (10)}) \\ &\geq \frac{1 - 2^{-k+1}}{2D_1} \prod_{i=1}^{k-2} \frac{1 - 2^{-i}}{1 - 2^{-i} + D_{i+1}} = \frac{1}{2} \prod_{i=1}^{k-1} \frac{1 - 2^{-i}}{1 - 2^{-i+1} + D_i}. \quad (11)\end{aligned}$$

Сначала оценим снизу величину $\prod_{i=1}^{k-1} (1 - 2^{-i})$. Имеем

$$\begin{aligned}\ln \prod_{i=1}^{k-1} (1 - 2^{-i}) &> \ln \prod_{i \geq 1} (1 - 2^{-i}) = \sum_{i \geq 1} \ln(1 - 2^{-i}) \\ &= - \sum_{i \geq 1} \sum_{j \geq 1} \frac{1}{j(2^i)^j} = - \sum_{j \geq 1} \sum_{i \geq 1} \frac{1}{j(2^j)^i} = - \sum_{j \geq 1} \frac{1}{j(2^j - 1)}.\end{aligned}$$

Так как $2j(2^j - 1) < (j + 1)(2^{j+1} - 1)$ при любом $j \geq 5$, то

$$- \sum_{j \geq 5} \frac{1}{j(2^j - 1)} > - \frac{2}{5(2^5 - 1)}.$$

Поэтому

$$\ln \prod_{i=1}^{k-1} (1 - 2^{-i}) > - \sum_{j=1}^4 \frac{1}{j(2^j - 1)} - \frac{2}{5(2^5 - 1)} > -1,25.$$

Следовательно,

$$\prod_{i=1}^{k-1} (1 - 2^{-i}) > e^{-1,25}. \quad (12)$$

Теперь оценим сверху величину

$$\prod_{i=1}^{k-1} (1 - 2^{-i+1} + D_i). \quad (13)$$

Нетрудно убедиться, что при фиксированной сумме величин D_i , равной N , максимум выражения (13) достигается в случае, когда все сомножители равны, т. е. при $D_1 = 1 - 2^{-1} + D_2 = 1 - 2^{-2} + D_3 = \cdots =$

$1 - 2^{-k+2} + D_{k-1}$, или $D_{i+1} = D_1 - 1 + 2^{-i}$, $i = 1, \dots, k-2$. (Здесь мы игнорируем ограничение $D_i \geq 1$.) При таких D_1, \dots, D_{k-1} имеем

$$N = \sum_{i=1}^{k-1} D_i = D_1(k-1) - (k-2) + \sum_{i=1}^{k-2} 2^{-i}.$$

Следовательно,

$$D_1 = \left(N + k - 2 - \sum_{i=1}^{k-2} 2^{-i} \right) / (k-1) < \frac{N + k - 2}{k-1},$$

а выражение (13) не превосходит $D_1^{k-1} \leq \left(\frac{N+k-2}{k-1} \right)^{k-1}$.

Нетрудно убедиться, что функция $\left(\frac{N-1+x}{x} \right)^x$ от x возрастает при $x > 0$, поскольку производная ее логарифма, равная $\ln \frac{N-1+x}{x} - \frac{N-1}{N-1+x}$, положительна. (Это следует из неравенства $\ln \frac{1}{1-\Delta} > \Delta$ при $\Delta \in (0, 1)$.)

Отсюда следует, что величина $\left(\frac{N+k-2}{k-1} \right)^{k-1}$ достигает максимума при $k = N$, т. е. выражение (13) не превосходит 2^{N-1} . Подставляя эту оценку, а также оценку (12) в (11), получим $\beta_k > e^{-1,25} \cdot 2^{-N}$. Таким образом, число больших работ не превосходит $n_{k-1} < 1/\beta_k < e^{1,25} \cdot 2^{m/\varepsilon}$.

Описанный выше алгоритм трудоемкости $O(n \log n)$ легко преобразуется в линейный. Поскольку для любого заданного примера гарантируется оценка $\alpha'' > \varepsilon / (e^{1,25} \cdot 2^N)$, то мы можем сначала за линейное время отобрать заведомо «малые» работы J_j , для которых выполняется неравенство $d_j \leq \frac{\varepsilon}{e^{1,25} \cdot 2^N} \hat{L}_{\max}$, а затем уже для оставшихся работ (число которых ограничено сверху константой $\frac{m}{\varepsilon} \cdot e^{1,25} \cdot 2^{m/\varepsilon}$) с помощью описанного выше алгоритма отыскивать числа α', α'' и распределение оставшихся работ по множествам L, M и S .

Таким образом, трудоемкость шага 2 есть $O(rn)$, а трудоемкость всего алгоритма \mathcal{A}_ε оценивается величиной $O(nrm)$, в которой от ε зависит лишь аддитивная константа. Теорема 1 доказана.

4. Об эффективности полиномиальной схемы

Рассмотрим пример с параметрами $m = 3$, $\varepsilon = 1/3$. Алгоритм, описанный в [4], гарантирует нахождение $4/3$ -приближенного решения задачи $O3||C_{\max}$ с линейной от n временной сложностью, где в качестве аддитивной константы в оценке сложности присутствует сложность нахождения приближенного решения задачи с 5 работами с абсолютной оценкой точности

$$C_{\max}(S) \leq \frac{4}{3} L_{\max}.$$

(Доказывается, что такое расписание существует для любых входных данных задачи $O3||C_{\max}$.) В описанной выше аппроксимационной схеме к линейной от n оценке сложности добавляется аддитивная константа, являющаяся верхней оценкой сложности построения расписания (причем оптимального) для существенно большего числа работ (при $m = 3$ и $\varepsilon = 1/3$ число «больших» работ может быть порядка тысячи). Таким образом, с практической точки зрения построенная нами схема оказывается неработоспособной.

Выход из этой ситуации состоит в дальнейшем совершенствовании схемы. В основу более совершенной схемы могли бы быть положены более глубокие знания о свойствах оптимальных расписаний задачи open shop. Так, например, мы знаем, что в любом плотном расписании количество внутренних интервалов простоя на любой машине не превосходит $r - 1$. И хотя для некоторых примеров может не существовать плотных оптимальных расписаний (один из таких примеров для трех работ на трех машинах — приведен в [5]), это не исключает возможности, что для одного из оптимальных расписаний любой конкретной задачи выполняется аналогичное свойство, т. е. количество внутренних интервалов простоя ограничено величиной, не зависящей от числа работ. Если бы такое свойство удалось доказать, мы могли бы предложить более эффективную аппроксимационную схему для рассматриваемой задачи, поскольку количество внутренних простоев в получаемом расписании не пришлось бы оценивать через число больших работ.

Авторы пользуются случаем поблагодарить А. Д. Коршунова, предложившего более короткое доказательство верхней оценки для числа больших работ.

ЛИТЕРАТУРА

1. Севастьянов С. В. Эффективное построение расписаний в системах открытого типа // Сиб. журн. исслед. операций. 1994. Т. 1, № 1. С. 20–42.
2. Lawler E. L., Lenstra J. K., Rinnooy Kan A. H. G., Shmoys D. B. Sequencing and scheduling: algorithms and complexity // Handbooks in Operations Research and Management Science. North-Holland: Elsevier, 1993. V. 4. P. 445–522.
3. Schuurman P., Woeginger G. J. Approximation algorithms for the multi-processor open shop problem // Report Woe-13. Oct. 1997. TU Graz, Austria.
4. Sevastianov S. V., Tchernykh I. D. Computer-aided way to prove theorems in scheduling // Algorithms — ESA'98. 6th Annu. European Symp. (Venice, Italy, Aug. 1998). Proc. Berlin: Springer, 1998. P. 502–513. (Lecture Notes in Comput. Sci.; V. 1461).

5. **Sevastianov S. V., Woeginger G. J.** Makespan minimization in open shops: A polynomial time approximation scheme // *Math. Programming.* 1998. V. 82, N 1–2. P. 191–198.
6. **Williamson D. P., Hall L. A., Hoogeveen J. A., Hurkens C. A. J., Lenstra J. K., Sevastianov S. V., Shmoys D. B.** Short shop schedules // *Oper. Res.* 1997. V. 45, N 2. P. 288–294.

Адреса авторов:

G. J. Woeginger

Institute für Mathematik,
TU Graz, Steyrergasse, 30,
A-8010 Graz, Austria.

E-mail:

gwoegi@opt.math.tu-graz.ac.at

С. В. Севастьянов

Институт математики
им. С. Л. Соболева СО РАН,
пр. Академика Коптюга, 4,
630090 Новосибирск, Россия.

E-mail: seva@math.nsc.ru

Статья поступила

5 ноября 1998 г.