ДИСКРЕТНЫЙ АНАЛИЗ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ
Июль—декабрь 2003. Серия 2. Том 10, № 2, 29–55

УДК 519.85

ИСПОЛЬЗОВАНИЕ ЧЕРЕДУЮЩИХСЯ ОКРЕСТНОСТЕЙ ДЛЯ ПРИБЛИЖЕННОГО РЕШЕНИЯ ЗАДАЧИ КАЛЕНДАРНОГО ПЛАНИРОВАНИЯ С ОГРАНИЧЕННЫМИ РЕСУРСАМИ*)

Ю. А. Кочетов, А. А. Столяр

Рассматривается известная NP-трудная задача календарного планирования с ограниченными ресурсами. Для ее решения предлагается новый алгоритм локального поиска, основанный на идее чередующихся окрестностей. Рассматриваются два типа дополняющих друг друга окрестностей. Одна из них строится по так называемым *активным* расписаниям, вторая окрестность — по T-поздним расписаниям. Окрестности имеют линейный размер относительно числа рассматриваемых работ и строятся с привлечением задачи о многомерном рюкзаке. Разработанный алгоритм тестировался на примерах из электронной библиотеки PSPLib. Для многих примеров алгоритм позволяет находить наилучшие уже известные решения, а для ряда наиболее трудных примеров – новые лучшие значения целевой функции.

Введение

В задаче календарного планирования с ограниченными ресурсами (ЗКПОР) задано множество работ, связанных друг с другом условиями предшествования. Эти условия задают технологию выполнения работ и на множестве работ порождают частичный порядок. Для каждой работы задана длительность ее выполнения и объемы потребляемых ресурсов. Суммарный объем каждого ресурса считается известным в каждый момент времени. Все ресурсы являются нескладируемыми [1], т. е. неиспользованные ресурсы пропадают как, например, рабочее время. Выделение и потребление ресурсов предполагается равномерным. Требуется

*)Исследование выполнено при финансовой поддержке Российского фонда фундаментальных исследований (проект 03–01–00455).

найти расписание выполнения работ, удовлетворяющее условиям предшествования, ограничениям по ресурсам и минимизирующее время окончания всех работ.

Сформулированная задача является NP-трудной. Для ее решения разработаны как точные методы, основанные на идеях метода ветвей и границ [11, 24], так и приближенные методы, базирующиеся на приоритетных правилах [5, 9, 18–20, 27, 32], лагранжевых релаксациях [25], идеях эволюции [16, 17, 34], локальном поиске [6, 10, 26, 28, 30, 33] и др. Обзор существующих методов для решения данной и близких к ней задач можно найти в монографии [35].

В настоящей статье предлагается новый алгоритм локального поиска, основанный на идее чередующихся окрестностей [15]. Рассматриваются два типа окрестностей. Первая окрестность строится по активным расписаниям, т. е. таким расписаниям, когда ни одна работа не может быть начата раньше указанного ей срока без нарушения либо условий предшествования, либо ограничений по ресурсам. Вторая окрестность определяется по T-поздним расписаниям. Такие расписания рассматривались в работе [1] для задач со складируемыми ресурсами и использовались для получения точных и асимптотически точных решений. В некотором смысле T-поздние расписания являются аналогом активных расписаний с той лишь разницей, что теперь каждая работа не может быть завершена позднее указанного ей срока без нарушения условий предшестования, ограничений по ресурсам, а также увеличения времени завершения проекта. Активные и T-поздние расписания удачно дополняют друг друга. Переход от одной окрестности к другой привносит определенное разнообразие в локальный поиск, что благотворно сказывается на результатах работы алгоритма.

Статья организована следующим образом. В разд. 1 дана математическая постановка задачи. В разд. 2 рассмотрены кодировки решений и декодирующие процедуры. Структура окрестностей приведена в разд. 3. В разд. 4 описана общая схема алгоритма. Результаты численных экспериментов приводятся в разд. 5. В последнем разделе содержатся выводы и направления дальнейших исследований.

1. Постановка задачи

Обозначим через $J = \{1, \ldots, n\} \cup \{0, n+1\}$ множество работ, в котором 0-я и $(n+1)$-я работы фиктивны и задают начало и завершение всего комплекса работ. Длительность j-й работы обозначим через $p_j,\ j \in J$. Считаем, что $p_j \geqslant 0,\ j \in J$ — целые неотрицательные числа и $p_0 = p_{n+1} = 0$. Отношения предшествования на множестве J зададим

множеством пар $C = \{(i,j) \mid i$ предшественник $j\}$. Если $(i,j) \in C$, то работа j не может начаться раньше завершения работы i. В множестве C содержатся все пары $(0,j)$ и $(j,n+1)$, $j = 1,\ldots,n$. Через P_j обозначим множество непосредственных предшественников работы j.

Через $K = \{1,\ldots,w\}$ обозначим множество ресурсов, необходимых для выполнения рассматриваемых работ. Все ресурсы предполагаются возобновляемыми и нескладируемыми, т. е. в каждый момент времени выделяется фиксированный объем ресурсов каждого типа, а остатки ресурсов пропадают. Будем считать, что объем выделяемого ресурса $R_k > 0, k \in K$, есть величина постоянная. Через $r_{jk} \geqslant 0$ обозначим объем k-го ресурса, необходимый для выполнения j-й работы.

Введем переменные задачи. Через $s_j \geqslant 0$ обозначим момент начала выполнения j-й работы. Предполагаем, что работы выполняются без прерывания, и момент окончания j-й работы определяется равенством $c_j = s_j + p_j$. Через $A(t) = \{j \in J \mid s_j \leqslant t < c_j\}$ обозначим множество работ, выполняемых в момент времени t. Время завершения проекта обозначим через $T(S)$. Эта величина соответствует моменту завершения последней работы проекта, т. е. $T(S) = c_{n+1}$. При этих обозначениях задача календарного планирования с ограниченными ресурсами записывается следующим образом.

Найти $\min T(S)$ при условиях

$$\begin{gathered} c_i \leqslant s_j,\ (i,j) \in C, \\ \sum_{j \in A(t)} r_{kj} \leqslant R_k,\ k \in K,\ t \geqslant 0, \\ s_j \geqslant 0,\ j \in J. \end{gathered}$$

Целевая функция задает время завершения всего комплекса работ. Первое неравенство задает условия предшествования. Второе неравенство требует выполнения ресурсных ограничений.

Сформулированная задача является NP-трудной в сильном смысле [8], так как одномерная задача упаковки в контейнеры является ее частным случаем. Более того, для любого $\varepsilon > 0$ маловероятно существование приближенных полиномиальных алгоритмов с гарантированной оценкой точности $n^{1-\varepsilon}$ [25]. Таким образом, разработка метаэвристик [29] является, по-видимому, наиболее перспективным направлением для решения данной задачи.

2. Кодировки и декодирующие процедуры

Кодировки решений имеют важное значение при разработке оптимизационных алгоритмов. Ниже приводится краткий обзор известных кодировок для ЗКПОР.

2.1. Кодировки решений

В большинстве работ, в которых изучаются алгоритмы решения ЗКПОР, используется кодировка решения в виде списка $L = (j_1, \ldots, j_n)$. Предполагается, что списки согласованы с условиями предшествования, т. е. для любой пары $(i, j) \in C$ позиция работы j больше позиции работы i. По списку можно построить расписание. Ниже будут приведены три алгоритма построения расписаний работ с использованием последовательного, параллельного и T-позднего декодеров. Первый декодер согласно списку последовательно вычисляет наиболее раннее время начала выполнения каждой работы, учитывая ресурсные ограничения. Второй декодер вычисляет наиболее раннее время, когда можно начать хотя бы одну из невыполненных работ и для этого момента времени принимает решение о начале выполнения очередных работ из списка. Произвол в выборе таких работ открывает широкие возможности для совершенствования декодеров этого типа, о чем пойдет речь в третьем разделе. T-поздний декодер действует аналогично первому, но использует список в обратном порядке. Фиксируется длина расписания T, и с учетом ресурсных ограничений для каждой работы вычисляется наиболее позднее время ее окончания. Как уже было сказано выше, условия предшествования учитываются при составлении списка и выполняются автоматически.

Для решения задач теории расписаний интенсивно исследовались быстрые полиномиальные алгоритмы, основанные на приоритетных правилах. Согласно этому подходу среди множества работ, доступных к выполнению, выбиралась одна работа по заданному критерию, например, работа с минимальной длительностью, минимальным числом предшественников, или другому критерию. Понятно, что ни одно из таких правил не гарантирует получения оптимального решения, если задача является NP-трудной. Поэтому более эффективной стратегией является применение сразу нескольких правил и использование одного из них в зависимости от уже построенного частичного расписания. Идея такой групповой стратегии применялась для ЗКПОР [9, 32]. Кодировка решений задается вектором $\pi = (\pi_1, \ldots, \pi_n)$, где π_i определяет приоритетное правило выбора очередной работы. Отметим, что такая кодировка, как и предыдущая, за полиномиальное время позволяет получить расписание работ, но одному расписанию может соответствовать несколько разных векторов π или списков L.

В работе [30] было предложено представление решений в виде вектора смещения. Решение задается вектором целых неотрицательных чисел

$\sigma = (\sigma_1, \ldots, \sigma_n)$. Декодирующая процедура последовательно вычисляет величины $s_j = \max\{s_i + p_i \mid i \in P_j\} + \sigma_j$, $j = 1, \ldots, n$. Поскольку декодирующая процедура игнорирует ограничения по ресурсам, полученное расписание может оказаться недопустимым. В этом случае к длине расписания добавляется величина штрафа, которая зависит от степени нарушения ресурсных ограничений.

Для переборных методов типа ветвей и границ была предложена специальная кодировка решений, ориентированная на анализ условий предшествования [11]. Эта кодировка называется *логической схемой*. Она представляет собой четверку непересекающихся отношений (C, D, N, F). Множество C задает исходные условия предшествования. Если $(i, j) \in D$, то работы i и j не могут пересекаться по времени. Если $(i, j) \in N$, то работы i и j должны выполняться параллельно в течение некоторого промежутка времени. Множество F содержит все оставшиеся пары (i, j), которые не противоречат множествам C, D и N. Конкретная четверка $(C,\ D,\ N,\ F)$ является кодом всех расписаний, в которых выполняются все отношения из $C,\ D$ и N.

2.2. Декодеры

Декодирующие процедуры строят расписание по заданной кодировке. Далее будет использоваться только кодировка в виде списка работ. Рассмотрим три декодера: последовательный, параллельный и T-поздний. Каждый декодер выполняет не более $n + 1$ шаг.

Последовательный декодер является наиболее простым из рассматриваемых [19]. На m-м шаге этой процедуры очередной работе в списке устанавливается момент начала ее выполнения как наиболее ранний из возможных с учетом условий предшествования и ограничений по ресурсам.

Последовательный декодер (L)

1. Полагается $s_0 := c_0 := 0$.
2. Для всех $m = 1, \ldots, n + 1$ проделывается следующее.

2.1. Для m-й работы в списке L находится

$$t_m = \max\{c_i \mid (i, j_m) \in C,\ i \in J\}.$$

2.2. Определяется минимальное время $t \geqslant t_m$ такое, что работа j_m может выполняться без нарушения ресурсных ограничений.

2.3. Полагается $s_{j_m} := t,\ c_{j_m} := t + p_{j_m}$.

3. Вычисляется $T(S) = c_{n+1}$.

Шаг 2.2 требует $O(nw)$ операций, если в каждый момент времени c_{j_m}

хранится объем использованных ресурсов. Таким образом, суммарная временная сложность процедуры не превосходит величины $O(n^2w)$.

Последовательный декодер строит расписание, в котором ни одна работа не может начаться раньше установленного ей срока без нарушений условий предшествования или ограничений по ресурсам. Такие расписания называются *активными*. Известно [19], что среди активных расписаний есть и оптимальное расписание.

Наряду с активными расписаниями будем рассматривать T-поздние расписания. Пусть T — время окончания $(n+1)$-й работы. Приведем алгоритм построения расписания, в котором ни одна работа не может закончиться позже установленного ей срока без нарушений либо условий предшествования, либо ограничений по ресурсам.

T-поздний декодер (L, T)

1. Полагается $s_{n+1} := c_{n+1} := T$.
2. Для всех $m = n, \ldots, 0$ проделывается следующее.

2.1. Для m-й работы в списке L находится момент

$$t_m = \min\{s_i \mid (j_m, i) \in C,\ i \in J\}.$$

2.2. Определяется наиболее позднее время $t \leqslant t_m$ такое, что работа j_m может закончиться без нарушения ресурсных ограничений.

2.3. Полагается $s_{j_m} := t - p_{j_m}$.

3. Вычисляется $T(S) = T - s_0$.

Сложность процедуры также оценивается величиной $O(n^2w)$. Если ограничения по ресурсам не зависят от времени, то среди T-поздних расписаний найдётся оптимальное. Доказательство аналогично случаю с активными расписаниями.

Параллельный декодер имеет существенное отличие от описанных выше. Если раньше последовательно рассматривались работы, и для каждой из них вычислялся момент начала её выполнения, то теперь последовательно рассматриваются возрастающие моменты времени, и для каждого из них определяются работы, начинающие выполняться в данный момент времени. Итак, на каждом шаге $m = 1, \ldots, n$ имеется некоторый момент времени t_m, множество $J(t_m) = \{j \in J \mid c_j \leqslant t_m\}$ уже завершенных работ и множество $A(t_m)$ работ, находящихся в процессе выполнения. По множеству $A(t_m)$ определяются остаточные объемы ресурсов $\widetilde{R}_k(t_m) = R_k - \sum\limits_{i \in A(t_m)} r_{ik}, k \in K$, и допустимое множество

$$D(t_m) = \{j \in J \setminus (A(t_m) \cup J(t_m)) \mid P_j \subseteq J(t_m),\ r_{jk} \leqslant \widetilde{R}_k(t_m), k \in K\}$$

работ, которые могут начаться в момент времени t_m. Шаг состоит в том, что из множества $D(t_m)$ выбирается работа с минимальной позицией в списке и она начинает выполняться в момент времени t_m. Формально параллельный декодер может быть представлен следующим образом.

Параллельный декодер (L)

1. Полагается $m := 0$, $t_m := 0$, $A(0) := \{0\}$, $J(0) := \emptyset$, $\widetilde{R}_k(0) := R_k$, $s_0 := c_0 := 0$.
2. Пока $|A(t_m) \cup J(t_m)| \leqslant n+1$ проделывается следующее.
   2.1. Полагается $m := m+1$ и $t_m = \min\{c_j \mid j \in A(t_{m-1})\}$.
   2.2. Вычисляется $J(t_m), A(t_m), \widetilde{R}_k(t_m), D(t_m)$.
   2.3. Пока $D(t_m) \neq \emptyset$ проделывается следующее:
      2.3.1. Выбирается из списка L работа $j \in D(t_m)$.
      2.3.2. Полагается $s_j := t_m$, $c_j := s_j + p_j$.
      2.3.3. Обновляется $A(t_m), \widetilde{R}_k(t_m), D(t_m)$.
3. Вычисляется $T(S) = c_{n+1}$.

Сложность данной процедуры оценивается величиной $O(n^2 w)$. Полученное расписание относится к классу *плотных* расписаний [19]. Для любого расписания легко проверить, является ли оно плотным. Для этого каждую работу j достаточно заменить на p_j работ единичной длительности и проверить, является ли полученное расписание активным. Известно [19], что класс плотных расписаний может не содержать оптимального расписания.

На шаге 2.3 последовательно выбираются работы из множества $D(t_m)$ до тех пор, пока хватает ресурсов. Более гибкой стратегией является выбор работ не по одной в соответствии со списком, а такой группой из множества $D(t_m)$, чтобы максимально использовать имеющиеся ресурсы $\widetilde{R}_k(t_m)$. Эта идея будет использована в следующем разделе при построении окрестностей.

3. Окрестности

Рассмотрим окрестности $N_A(S)$ и $N_T(S)$. Первая из них строится для активных расписаний, вторая — для T-поздних. Эти окрестности имеют линейный размер, а для определения соседних решений используют параллельный декодер.

3.1. Окрестность $N_A(S)$ для активного расписания S

Определение 1. *Блоком* работы j в активном расписании S называется множество работ, которые выполняются параллельно с работой j, либо начинают выполняться сразу за работой j, либо заканчиваются

непосредственно перед ней: $B_j = \{i \in J \mid [s_i, c_i] \cap [s_j, c_j] \neq \emptyset\}$.

Наряду с блоком введем понятие исходящей сети. Пусть $G_S(V, E)$ — орграф с множеством вершин $V = J$ и множеством дуг

$$E = \{(i, j) \mid c_i = s_j,\ (i, j) \in C\}.$$

Определение 2. *Исходящей сетью* работы j для расписания S называется максимальный по включению связный подграф графа G_S, в котором единственным источником является вершина, соответствующая работе j.

Исходящая сеть позволяет найти все работы в расписании S, которые по условиям предшествования не могут начаться раньше указанного им срока, если время выполнения работы j не меняется.

Пусть активное расписание S получено по списку L последовательным декодером, а блок работы j не содержит ее предшественников, т. е. $B_j \cap P_j = \emptyset$. Выделим множество работ, время выполнения которых следует оптимизировать, если меняются сроки выполнения работы j. Это множество определим по сегменту $L(j)$ списка L. Началом сегмента является работа блока B_j с минимальной позицией в списке L. Конец сегмента определяется как максимум из двух чисел: максимальной позиции в списке L для работ блока B_j и исходящей сети. Итак, по работе j выделен сегмент $L(j)$, для работ которого будут определяться новые сроки их выполнения.

Представим список L в виде трех последовательных списков $L = L', L(j), L''$. Для работ списка из L' сохраним старые сроки их выполнения, а для работ из списка $L(j)$ будем применять модификацию параллельного декодера, в которой на шаге 2.3 будет выбираться подмножество работ из допустимого множества $D(t_m)$ по критерию максимизации использованных ресурсов [34]. Введем переменные $x_i \in \{0, 1\},\ i \in D(t_m)$. Если $x_i = 1$, то для i-й работы из множества $D(t_m)$ положим $s_i := t_m$. Если же $x_i = 0$, то $s_i > t_m$. Формально, выбор нужного подмножества означает решение следующей задачи о многомерном рюкзаке. Найти

$$\max_{x_i \in \{0,1\}} \sum_{i \in D(t_m)} x_i \sum_{k \in K} \frac{r_{ik}}{R_k}$$

при ограничениях $\sum\limits_{i \in D(t_m)} r_{ik} x_i \leqslant R_k - \sum\limits_{i \in A(t_m)} r_{ik},\ k \in K.$

Целевая функция задачи требует максимизации суммарной доли использованных ресурсов в момент времени t_m. Ограничения задачи устанавливают границы потребления ресурсов. Для решения данной задачи

применяется вероятностный жадный алгоритм, о котором пойдет речь в разделе 3.3.

С помощью указанной модификации параллельного декодера получаем новое расписание работ из списка $L(j)$. Последовательным декодером частичное расписание достраивается до расписания $S_A(j)$ всех работ множества J. Расписание $S_A(j)$ назовем *соседним* расписанием для S. Множество всех соседних расписаний назовем *окрестностью* расписания S и обозначим через $N_A(S)$.

3.2. Окрестность $N_T(S)$ для T-позднего расписания S

Определение 3. *Входящей сетью* работы j для T-позднего расписания S называется максимальный по включению связный подграф графа G_S, в котором единственным стоком является вершина, соответствующая работе j.

Пусть расписание S получено по списку L применением T-позднего декодера, и блок работы j не содержит ее последователей, т. е.

$$B_j \cap P_i = \emptyset, (j,i) \in C.$$

По аналогии с определением окрестности $N_A(S)$ выделим сегмент $L(j)$ в L. Началом сегмента является минимальная позиция в списке L для работ блока B_j и входящей сети. Концом сегмента является максимальная позиция в L работ блока B_j.

Пусть, как и прежде, $L = L', L(j), L''$. Сохраним старые сроки выполнения работ из списка L''. Применим к нему модификацию параллельного декодера, которая в обратном порядке вычисляет времена выполнения работ из сегмента $L(j)$, используя задачу о многомерном рюкзаке. Для оставшихся работ из L' времена выполнения определяются T-поздним декодером. Полученное расписание назовем *соседним* для S и обозначим через $S_T(j)$. Множество всех соседних расписаний назовем *окрестностью* T-позднего расписания S и обозначим через $N_T(S)$.

3.3. Задача о многомерном рюкзаке

При определении окрестностей $N_A(S)$ и $N_T(S)$ существенно использовалась задача о многомерном рюкзаке. Известно, что она NP-трудна даже при $w = 1$. В связи с этим рассмотрим вероятностный жадный алгоритм [2], который позволяет быстро получать для нее приближенные решения.

Каждое ограничение разделим на $R_k,\ k \in K$. При $w = 1$ получаем одинаковые коэффициенты в целевой функции и ограничениях, т. е.

удельные стоимости одинаковы и равны 1. Если $w > 1$, то удельных стоимостей несколько, но их сумма равна 1:

$$\sum_{k\in K}(r_{ik}/R_k)\Big/\Big(\sum_{k'\in K} r_{ik'}/R_{k'}\Big) = 1.$$

Таким образом, для решения возникающей задачи о многомерном рюкзаке целесообразно использовать аналог жадных алгоритмов GS и MTGS (см. [22], гл. 4), разработанных для задачи о камнях. Эти алгоритмы используют только коэффициенты целевой функции при выборе очередного элемента.

Зададимся величиной $0 < q \leqslant 1$ и сформируем случайное подмножество $D(q) \subseteq D(t_m)$. Каждый элемент из $D(t_m)$ включается в множество $D(q)$ с вероятностью q независимо от других элементов.

Найдем в $D(q)$ элемент с максимальным весом $\sum_{k\in K} r_{ik}/R_k$, $i \in D(q)$. Значение соответствующей переменной x_i положим равным 1, уменьшим правые части ограничений и удалим из $D(t_m)$ выбранный элемент и все элементы, которые уже не помещаются в "рюкзак". Эту процедуру будем повторять до тех пор, пока множество $D(t_m)$ не станет пустым. На выходе имеем набор значений булевых переменных $x_i \in \{0,1\}$, $i \in D(t_m)$. Формально этот алгоритм может быть представлен следующим образом.

Алгоритм MP$(q, t_m, A(t_m), D(t_m))$

1. Полагается $\overline{D} := D(t_m)$, $\widetilde{R}_k := R_k - \sum\limits_{i\in A(t_m)} r_{ik}$, $x_i := 0$, $i \in \overline{D}$.
2. Выбирается в $\overline{D}$ случайным образом подмножество $D(q)$.

   Если $D(q) = \emptyset$, то к $D(q)$ присоединяется любой элемент из $\overline{D}$.
3. Находится $i_0 \in D(q)$ с максимальным весом

$$\sum_{k\in K} r_{i_0k}/R_k = \max_{i\in D(q)} \sum_{k\in K} r_{ik}/R_k.$$

4. Полагается $x_{i_0} := 1$; $\overline{D} := \overline{D} \setminus \{i_0\}$, $\widetilde{R}_k := \widetilde{R}_k - r_{i_0k}$, $k \in K$.
5. Для всех $i \in \overline{D}$: если $r_{ik} > \widetilde{R}_k$ для некоторого $k \in K$, то полагается $\overline{D} := \overline{D} \setminus \{i\}$.
6. Если $\overline{D} \neq \emptyset$, то переход на шаг 2.

Временная сложность алгоритма не превосходит величины

$$O(|D(t_m)|(w + |D(t_m)|)).$$

Алгоритм MP можно применять несколько раз и лучшее из полученных решений использовать при построении окрестностей $N_A(S)$ и $N_T(S)$. Ве-

личина q является параметром алгоритма. При $q = 1$ получаем детерминированный алгоритм GS. При малых значениях q алгоритм многократно формирует множество $D(q)$ и тем самым использует идею алгоритма MTGS, игнорируя некоторые элементы с большими весами.

4. Общая схема

Разработанный алгоритм сочетает в себе идеи двух методов: локального поиска с чередующимися окрестностями [15] и поиска с запретами [14]. За основу принята схема поиска с запретами, в которой систематически осуществляется переход от окрестности $N_A(S)$ к $N_T(S)$ и наоборот.

4.1. Начальное решение

Хорошее начальное приближение позволяет выбрать перспективную область и сконцентрировать усилия на улучшении уже достаточно хорошего решения. Выбор начального приближения не является критическим для методов локального поиска, но неудачный выбор может потребовать неоправданных затрат на исправление ситуации.

Среди допустимых списков выберем список L_0, в котором для любой соседней пары работ (j_m, j_{m+1}), не принадлежащей множеству C, справедливо неравенство $\sum\limits_{k\in K} r_{j_m k}/R_k \geqslant \sum\limits_{k\in K} r_{j_{m+1} k}/R_k$. Множество таких списков непусто. Один из них можно построить, например, упорядочив работы по рангам в сети, порожденной частичным порядком C, а работы с равными рангами — по весу.

Применим к списку L_0 рандомизированный вариант параллельного декодера. На шаге 2.3 вместо множества $D(t_m)$ будем использовать его непустое подмножество, выбранное случайным образом. В результате получим некоторое расписание S_0. Далее будут применяться T-поздний и последовательный декодеры до тех пор, пока это приводит к убыванию целевой функции.

Положим $T = T(S_0)$ и по расписанию S_0 построим список L_1, упорядочив работы по времени их окончания, $c_{j_0} \leqslant c_{j_1} \leqslant \ldots \leqslant c_{j_{n+1}}$. К полученному списку применим T-поздний декодер. Получим расписание S_1. Построим список L_2, упорядочив работы по началу их выполнения $s_{j_0} \leqslant s_{j_1} \leqslant \ldots \leqslant s_{j_{n+1}}$, и применим последовательный декодер. Получим расписание S_2. Если $T > T(S_2)$, то повторим процедуру с T-поздним и последовательным декодерами. Неформально, эта процедура напоминает игру в пинг-понг. Случайным образом формируется начальное расписание. Затем идет обмен между активными и T-поздними расписаниями

до тех пор, пока удается сокращать время выполнения всех работ. Формально этот алгоритм может быть представлен следующим образом.

Алгоритм Пинг-понг

1. Строится начальный список и к нему применяется рандомизированный вариант параллельного декодера.
2. Полагается $T := s_{n+1}$.
3. Работы упорядочиваются так, что $c_{j_0} \leqslant c_{j_1} \leqslant \ldots \leqslant c_{j_{n+1}}$, и строится T-позднее расписание.
4. Работы упорядочиваюся так, что $s_{j_0} \leqslant s_{j_1} \leqslant \ldots \leqslant s_{j_{n+1}}$, и строится активное расписание.
5. Если $T > s_{n+1}$, то полагается $T := s_{n+1}$ и переход к шагу 3.

Каждый шаг этой процедуры является полиномиальным по числу выполняемых операций. Однако число возвращений на шаг 3 может оказаться очень большим. Оценить это число сверху полиномом от длины записи исходных данных не представляется возможным. По-видимому, этот вопрос тесно связан с другим, более широким вопросом о сложности поиска локальных оптимумов для трудных комбинаторных задач [36]. Тем не менее, численные эксперименты показывают малое число возвращений на шаг 3. Как правило, это число значительно меньше n.

4.2. Вероятностный поиск с запретами

Приведем общую схему вероятностного алгоритма поиска с запретами. На каждом шаге этой процедуры имеется некоторое расписание S и значение функции $f(S) = \sum\limits_{j \in J} s_j$ от расписаний, полученных на последних h шагах алгоритма. Набор таких значений будем называть списком запретов. Шаг алгоритма состоит в переходе от расписания S к соседнему расписанию S' по окрестности $N_A(S)$ или $N_T(S)$. Для окрестности формируется случайным образом ее непустое подмножество $N'_A(S)$ или $N'_T(S)$, из которого выбирается расписание с минимальным значением целевой функции. Подмножество $N'_A(S)$ ($N'_T(S)$) формируется следующим образом. Из окрестности $N_A(S)$ ($N_T(S)$) удаляются все расписания, в которых значение функции $f(S)$ совпадает с одним из значений в списке запретов. Затем каждое из оставшихся расписаний включается в множество $N'_A(S)$ ($N'_T(S)$) с некоторой вероятностью q. Если длина списка запретов велика, то окрестность может оказаться пустой в результате удаления "запрещенных" элементов. В этом случае длина списка запретов уменьшается. Если подмножество $N'_A(S)$ ($N'_T(S)$) оказалось пустым, то в него добавляется произвольный незапрещенный элемент из окрестности $N_A(S)$ ($N_T(S)$).

Алгоритм поиска с запретами

1. Строится начальное расписание S,
   полагается $T^* := T(S),\ S^* := S$.
2. Пока не выполнен критерий остановки проделывается следующее.
   2.1. Выбирается окрестность.
   2.2. Находится соседнее расписание S'.
   2.3. Если $T(S') < T^*$, то полагается $T^* := T(S'),\ S^* := S'$.
   2.4. Обновляется список запретов и полагается $S := S'$.

Значение T^* является результатом работы алгоритма. Оно соответствует наилучшему расписанию S^*, найденному в ходе работы алгоритма. В качестве критерия остановки используется либо максимальное число итераций, либо требуемое отклонение от заданной верхней или нижней оценки целевой функции. Смена окрестности производится через заданное число итераций. Коэффициент рандомизации q в ходе поиска не меняется.

Приведенная схема алгоритма является одной из наиболее простых и может быть дополнена правилами интенсификации и диверсификации поиска [14]. Если значение T^* не меняется на нескольких последовательных итерациях, то целесообразно вернуться к расписанию S^* и более тщательно исследовать эту часть допустимой области, либо выбрать новое начальное расписание.

Отметим, что список запретов, кроме расписаний, полученных на последних итерациях, не допускает многие другие расписания. В работах [6, 26, 31] рассматривались другие списки запретов. Они соответствовали окрестностям, полученным сдвигом одной или нескольких работ в списке L. Список запретов препятствовал возвращению работ на прежнее место и тем самым предотвращал зацикливание алгоритма. Однако, как отмечалось выше, одному расписанию может соответствовать несколько списков и расписание может остаться прежним в результате сдвига нескольких работ. Функция $f(S)$ устраняет этот недостаток, хотя и возникает опасность запрета всех соседних расписаний, особенно на задачах малой размерности. В связи с этим величина h не должна быть слишком большой и ее изменение должно тщательно контролироваться (см., например, [3, 7]).

5. Численные эксперименты

Для тестирования алгоритмов решения задач календарного планирования с ограниченными ресурсами создана специализированная библиотека тестовых задач PSPLib [21]. В ней содержится много примеров

разной вычислительной трудности для рассматриваемой модели и для других моделей календарного планирования. Для ЗКПОР в этой библиотеке имеется по 480 примеров для размерности n=30, 60 и 90 работ и 600 примеров для n=120. Число типов ресурсов равно 4. Примеры разбиты на классы. В каждом классе имеется 10 примеров, порожденных с помощью датчика псевдослучайных чисел при фиксированных значениях трех параметров:

- $NC \in \{1,5; 1,8; 2,1\}$ – среднее число непосредственных предшественников каждой работы.
- $RF \in \{1; 2; 3; 4\}$ – число типов ресурсов, необходимых для выполнения каждой работы.
- $RS \in \{0,2; 0,5; 0,7; 1,0\}$ – объем выделяемых ресурсов в каждый момент времени; значение $RS = 0,2$ соответствует примерам с минимальным объемом выделяемых ресурсов, достаточным для разрешимости задачи; значение $RS = 1$ соответствует случаю неограниченных ресурсов.

Известно [35], что значения параметров $RF = 4$, $RS = 0,2$ соответствуют достаточно трудным классам. Их индентификаторы j3013, j3029, j3045 при $n = 30$; j6013, j6029, j6045 при $n = 60$; j12016, j12036, j12056 при $n = 120$ соответствуют значениям $NC = 1,5; 1,8; 2,1$. Для этих классов проводилась основная часть численных экспериментов.

5.1. Эффект чередующихся окрестностей

Для методов локального поиска выбор окрестности играет решающую роль. Интуитивно кажется, что чем шире окрестность, тем эффективнее поиск: шире области притяжения, меньше значения локальных минимумов. Конечно, для более широкой окрестности тратится больше времени на ее просмотр, но если на это обстоятельство не обращать внимания, то результаты поиска должны быть лучше. Однако, это не всегда так. Широкая область притяжения препятствует переходу от одного локального оптимума к другому. Эффективность методов падает, так как поиск концентрируется в малой части допустимой области.

Для преодоления этой трудности в работе [15] была предложена простая, но эффективная стратегия. Она использует не одну большую окрестность, а несколько окрестностей разной структуры и размера. Переход от одной окрестности к другой меняет ландшафт, области притяжения локальных оптимумов и привносит элемент диверсификации без кардинальной смены области поиска.

Т а б л и ц а 1

Изменение $\varepsilon(\%)$ при различном выборе окрестности

Класс	N_A	N_T	$N_A,\ N_T$	$N_A \cup N_T$
j3013	0,65	0,67	0,07	0,00
j3029	0,87	0,45	0,11	0,13
j3045	0,35	0,53	0,08	0,00
j6013	1,60	1,30	0,83	0,41
j6029	1,69	1,43	0,81	0,85
j6045	0,85	0,82	0,27	0,66
j12016	2,15	1,99	1,42	0,71
j12036	1,97	1,74	1,13	0,91
j12056	1,91	1,85	1,03	1,16

Первый численный эксперимент связан с влиянием чередования окрестностей на эффективность предложенного метода. Таблица 1 содержит среднюю относительную погрешность ε решения тестовых задач с числом работ 30, 60 и 120 для четырех вариантов поиска:

- только по окрестности $N_A(S)$,
- только по окрестности $N_T(S)$,
- с чередованием окрестностей $N_A(S)$ и $N_T(S)$.
- с объединением окрестностей $N_A(S) \cup N_T(S)$.

При n=30 погрешность считалась относительно точного решения. При n>30 для подсчета погрешности вместо точного решения использовалось наилучшее известное решение из библиотеки PSPLib.

Чередование окрестностей положительно сказывается на результатах поиска. Интересно отметить, что расширение окрестности до объединения $N_A(S) \cup N_T(S)$ не лучше их чередования. Последняя колонка в таблице 1 свидетельствует, что объединение окрестностей не всегда приводит к сокращению погрешности. На рис. 1 показано влияние интервала чередования окрестностей. При больших интервалах ($\tau > 1000$) по сути получаем последовательный поиск по окрестностям $N_A(S)$ и $N_T(S)$. Оптимальный интервал чередования соответствует 5–10 шагам алгоритма.

Для первых трех вариантов алгоритма помимо относительной погрешности была подсчитана частота нахождения лучшего известного решения.

Рис. 1. Влияние интервала чередования окрестностей, n=60

Частота представляет собой статистическую оценку вероятности получения лучшего решения. Для данной оценки построены доверительные интервалы [2]. Рассмотрим схему Бернулли с вероятностью успеха p. Под успехом будем понимать случайное событие, состоящее в том, что алгоритм нашел наилучшее известное решение. Рассмотрим серию из N испытаний и набор булевых переменных $x_i,\ i = 1, \ldots, N$. Полагаем $x_i = 1$, если в i-м испытании алгоритм нашел лучшее решение, и $x_i = 0$ в противном случае. Тогда случайная величина $p^* = \overline{X} = \sum x_i/N$ есть частота получения алгоритмом лучшего решения. Заметим, что величина $\frac{\sqrt{N}(\overline{X}-p)}{\sqrt{\overline{X}(1-\overline{X})}}$ слабо сходится к стандартному нормальному закону. Отсюда доверительный интервал для параметра $p^* = \overline{X}$ имеет вид

$$\overline{X} - \frac{\tau_{1-\alpha/2}\sqrt{\overline{X}(1-\overline{X})}}{\sqrt{N}} \leqslant p \leqslant \overline{X} + \frac{\tau_{1-\alpha/2}\sqrt{\overline{X}(1-\overline{X})}}{\sqrt{N}},$$

где $\tau_{1-\alpha/2}$ – квантиль стандартного нормального распределения уровня $1 - \alpha/2$. При $\alpha = 0,05$ значение квантили составляет 1,96. Значения доверительных интервалов приведены в таблице 2. Из таблицы видно, что в большинстве рассмотренных классов доверительные интервалы, соответствующие алгоритму с одной окрестностью и с чередующимися окрестностями, не пересекаются.

Т а б л и ц а 2

Доверительные интервалы для частоты получения лучшего решения, $\alpha = 0,05,\ N = 100$

Класс	N_A	N_T	N_A, N_T
j3013	[0,63 ; 0,81]	[0,56 ; 0,74]	[0,87 ; 0,97]
j3029	[0,39 ; 0,59]	[0,60 ; 0,78]	[0,82 ; 0,94]
j3045	[0,68 ; 0,84]	[0,64 ; 0,72]	[0,87 ; 0,97]
j6013	[0,00 ; 0,08]	[0,05 ; 0,17]	[0,19 ; 0,37]
j6029	[0,01 ; 0,09]	[0,06 ; 0,18]	[0,15 ; 0,31]
j6045	[0,20 ; 0,38]	[0,26 ; 0,44]	[0,41 ; 0,61]
j12016	[0,00 ; 0,00]	[0,00 ; 0,00]	[0,00 ; 0,05]
j12036	[0,00 ; 0,03]	[0,00 ; 0,00]	[0,01 ; 0,11]
j12056	[0,00 ; 0,00]	[0,00 ; 0,03]	[0,00 ; 0,08]

5.2. Размер окрестности и ее рандомизация

Следующий численный эксперимент связан с выбором коэффициента q рандомизации окрестности. При малых q время просмотра окрестности сокращается, но появляется опасность пропустить оптимальное решение, даже находясь в его окрестности. При q=1 просматриваются все соседние решения, но так как отбор лучшего решения в окрестности ведется по целевой функции, а она меняется незначительно, то, как правило, всегда находятся соседние решения с нехудшим значением функционала. По сути эти решения мало отличаются от текущего, идет перестановка некритических работ и эффективность поиска падает. "Золотая середина" в данном случае соответствует $q \approx 0,2$ (см. рис. 2.). Близкие значения рандомизации наблюдались и для задач размещения [3]. По-видимому, положительный эффект рандомизации при локальном поиске имеет место и для других трудных комбинаторных задач.

Возвращаясь к вопросу об оптимальной мощности окрестности, был проведен еще один эксперимент. При определении соседних решений рассматривалась задача о многомерном рюкзаке и для нее вместо точного алгоритма применялся вероятностный жадный алгоритм. Его достоинство состоит не только в том, что он позволяет быстро получать приближенные решения задачи. Не менее важным является возможность получать различные приближенные решения. Ее можно использовать по-разному. Например, многократно применять алгоритм и запоминать лучшее из полученных решений. Таким образом повышается точность решения за счет увеличения временной сложности. Правда, ниоткуда не

следует, что чем точнее решение, тем меньше окажется длина расписания. Поэтому логично рассматривать не только наилучшее решение для

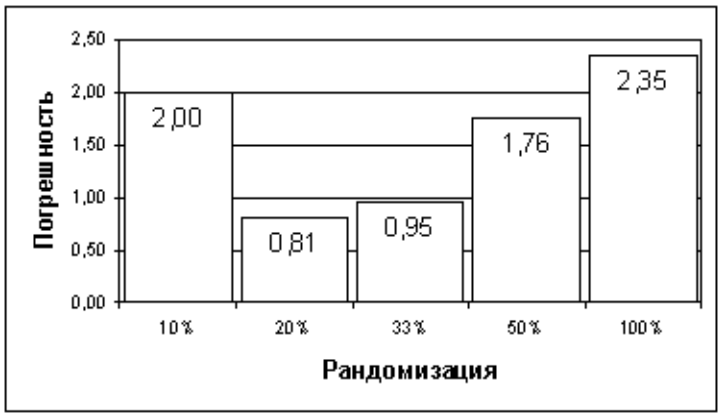

Рис. 2. Влияние рандомизации, класс j6029

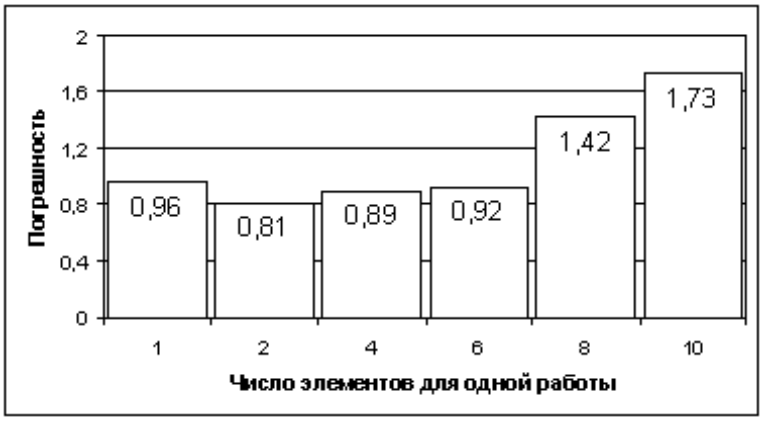

Рис. 3. Влияние размера окрестности, класс j6029, q=0,2

задачи о многомерном рюкзаке, но и другие близкие к нему решения В ходе численного эксперимента варьировалось число лучших приближенных решений, отбираемых при решении задачи о рюкзаке, и для каждого их них строилось соседнее расписание. Таким образом, размер окрестности возрастал в несколько раз. Рис. 3. показывает влияние этого процесса на относительную погрешность алгоритма. Добавление "плохих" приближенных решений задачи о рюкзаке не дает положительного эффекта, засоряет окрестность некачественными расписаниями и ухудшает результаты поиска. Тем не менее лучший выбор соответствует не

одному, а двум приближенным решениям задачи о многомерном рюкзаке.

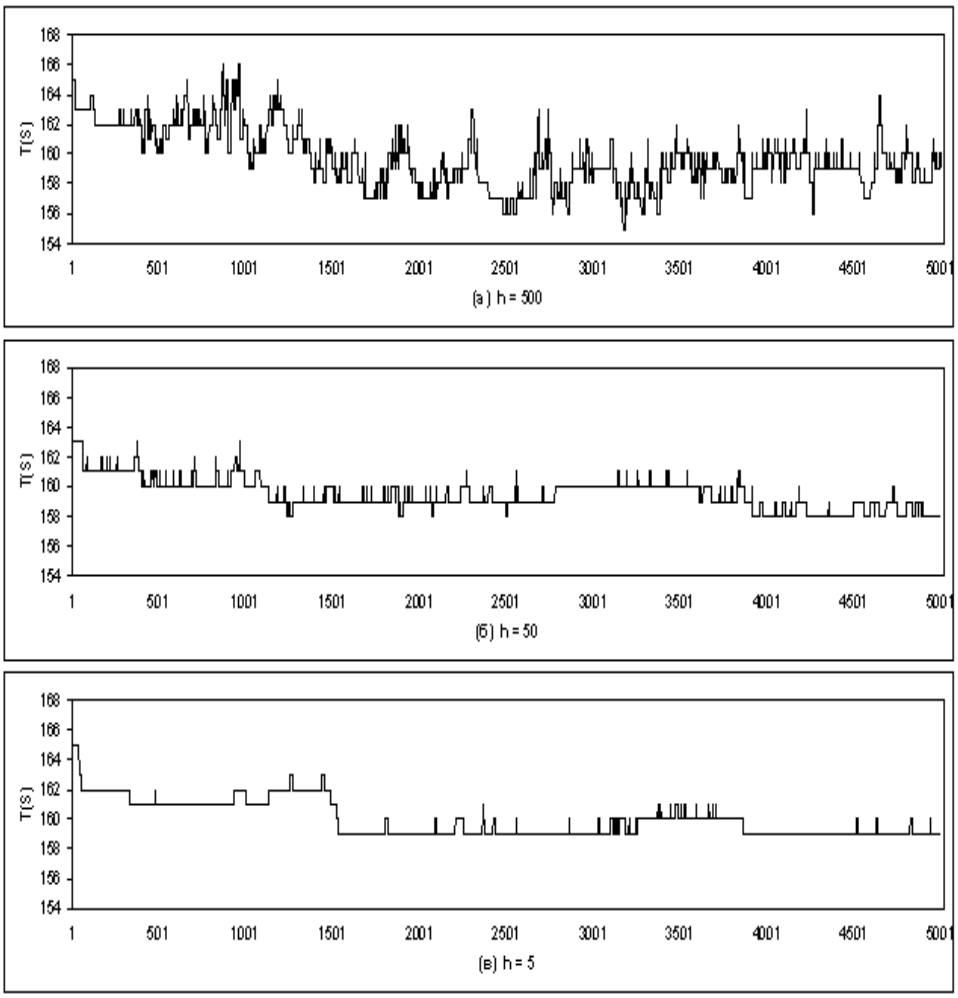

Рис. 4. Влияние списка запретов, j6029, $q = 1$

5.3. Влияние списка запретов

Основным назначением списка запретов является предотвращение зацикливания. Как уже отмечалось в п. 4.2., функция $f(S) = \sum\limits_{j \in J} s_j$ запрещает не только просмотренные решения, но и многие другие. Целью следующего эксперимента было определение оптимальной длины списка запретов и его влияния на относительную погрешность. На рис. 4. показано типичное поведение алгоритма при длине списка запретов h=5, 50

и 500. Наличие горизонтальных участков (см. рис. 4. б) и в)) свидетельствует о недостаточной длине списка. Алгоритм попадает на какое-то

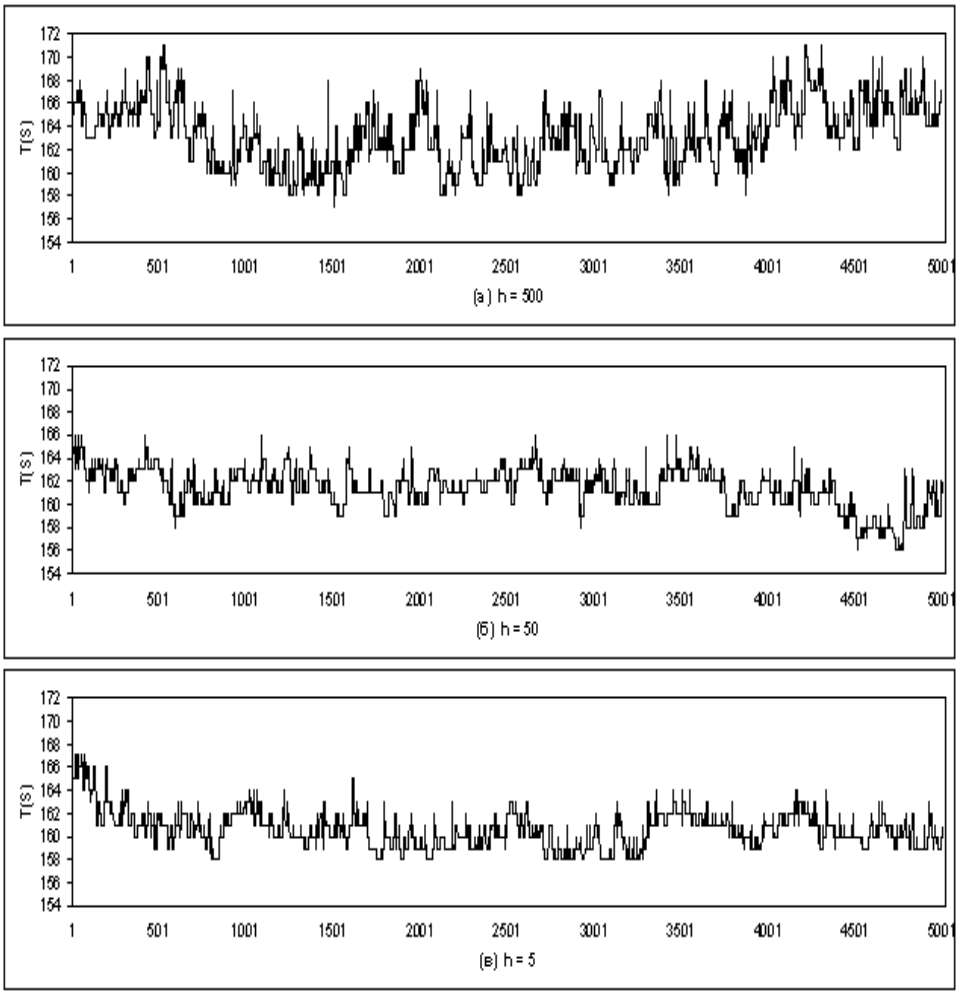

Рис. 5. Влияние списка запретов, j6029, $q = 0,2$

"плато" и длительное время не может с него сойти. При $h = 500$ такого эффекта нет и поиск действительно более продуктивен.

Внесение рандомизации окрестности меняет поведение алгоритма. При $q = 0,2$ (см. рис. 5.) уже при h=5 "плато" не является препятствием для алгоритма. Влияние списка запретов ослабевает. Таким образом, рандомизация окрестности не только сокращает время ее просмотра, но и позволяет хранить и проверять списки запретов меньшей длины.

5.4. Выбор начального решения

Следующий эксперимент связан с выбором начального решения. В таблице 3 приведены значения относительной погрешности в случае когда начальные решения выбираются случайно, а также с помощью алгоритма "Пинг-понг".

Т а б л и ц а 3
Влияние начального решения на $\varepsilon(\%)$

Класс	Случайное	Пинг-понг
j3013	0,00	0,07
j3029	0,12	0,11
j3045	0,10	0,08
j6013	0,81	0,83
j6029	0,99	0,81
j6045	0,50	0,27
j12016	1,47	1,42
j12036	1,22	1,13
j12056	1,22	1,03

Как видно из таблицы 3, для задач малой размерности выбор начального решения почти не влияет на итоговый результат. Преимущество в выборе хорошего начального решения сказывается с ростом размерности задачи. Это объясняется тем, что в случае небольших задач локальному поиску требуется незначительное число итераций для существенного улучшения неудачно выбранного начального приближенного решения. В задачах большой размерности требуется большее время для исправления ситуации. Несмотря на то, что при n=120 алгоритм выполнял 20000 шагов, случайный выбор начального решения привел к худшему результату на всех трех классах j12016, j12036 и j12056. Таким образом, для решения задач большой размерности целесообразно использовать эвристические процедуры при генерации начального приближения.

5.5. Интенсификация поиска

Последний эксперимент связан с влиянием интенсификации поиска на поведение алгоритма. В этом эксперименте сравниваются три варианта алгоритма, различающихся частотой возвращения к наилучшему найденному решению:

- поиск без возвращения с общим числом итераций $I(n)$,
- поиск с возвращением через каждые $I(n)/5$ итераций,

- поиск с возвращением через каждые $I(n)/10$ итераций.

Общее число итераций $I(n)$ составляло 5000, 10000 и 20000 при $n = 30, 60$ и 120 соответственно. В таблице 4 приведены средние значения ε_1, ε_5 и ε_{10} относительной погрешности для каждого из трех вариантов алгоритма.

Т а б л и ц а 4
Влияние интенсификации поиска, $q = 0,2$

Класс	ε_1	ε_5	ε_{10}
j3013	0,07	0,07	0,10
j3029	0,22	0,11	0,20
j3045	0,08	0,08	0,12
j6013	0,92	0,83	0,84
j6029	0,96	0,81	0,86
j6045	0,26	0,27	0,27
j12016	1,47	1,42	1,48
j12036	1,20	1,13	1,11
j12056	0,94	1,03	1,16

Как видно из таблицы 4, второй вариант алгоритма имеет небольшое преимущество в большинстве рассмотренных случаев. Это позволяет сделать вывод о том, что интенсификация поиска не оказывает существенного влияния на результат работы алгоритма.

5.6. Сравнение с другими алгоритмами

Результаты сравнения разработанного алгоритма поиска с запретами и чередованием окрестностей (ПЗЧО) с другими известными алгоритмами приведены в таблице 5. Алгоритмы сравнивались по двум показателям: средняя относительная погрешность и время счета. Для сравнения рассмотрены следующие алгоритмы:

- Гибридный алгоритм (ГИА) [28] сочетает в себе идею локального поиска с элементами полного перебора. Такой подход, по-видимому, обеспечивает высокое качество получаемых решений, но требует достаточно много времени.
- Эволюционный алгоритм (ЭА) [34], основанный на стратегии связывающих путей. В этом алгоритме по сути отсутствует локальный поиск, что приводит к малому времени счёта.

- Генетический алгоритм (ГА) [17], использующий кодировку списком и двухточечный оператор скрещивания. Особенностью этого алгоритма является то, что в нем на каждой итерации проводится случайный выбор между последовательным и параллельным декодером.
- Алгоритм имитации отжига (ИО) [10], использующий списочную кодировку и последовательный декодер.
- Алгоритм поиска с запретами (ПЗ1) [6], использующий кодировку в виде логической схемы со специально разработанным для нее декодером.
- Алгоритм поиска с запретами (ПЗ2) [26], использующий списочную кодировку и последовательный декодер.
- Поиск по переменной окрестности (ППО) [13], использующий списочную кодировку и последовательный декодер.
- Алгоритм муравьиной колонии (МК) [13], использующий последовательный декодер, ориентированный на приоритетное правило LFT (Latest Finish Time).
- Алгоритм лагранжевой релаксации (ЛР) [25], использующий задачу о минимальном разрезе при построении нижней оценки.
- Алгоритм 'CARA' [33], представляющий собой локальный поиск, основанный на перемещении критических работ.

В [31] предложен оригинальный вариант алгоритма поиска с запретами. Он не использует кодировок, а работает прямо с расписаниями. К сожалению, алгоритм тестировался на тестовых задачах из другой библиотеки. Однако известно, что он уступает генетическому алгоритму, предложенному в [17].

Таблица 5 имеет следующую структуру. В первой колонке стоит название алгоритма. Во второй, третьей и червертой — числовые характеристики алгоритмов для задач размерности 30, 60 и 120 соответственно. Каждая из трех последних колонок разбита на две подколонки, в которых соответственно представлены средние значения относительной погрешности и времени счета для всех примеров данной размерности из библиотеки PSPLib. Для задач с числом работ, равным 30, погрешность вычислялась по отношению к оптимуму. Для задач большей размерности данная величина вычислялась по отношению к нижней оценке, полученной с помощью метода критического пути. Алгоритмы тестировались на следующих вычислительных машинах:

ПЗЧО — PENTIUM III 1800MHz 256Mb RAM;
ЭА, CARA — AMD 400MHz;
ППО — PENTIUM III 1GHz;
ГИА — 4 proc. HP 9000 440MHz 2Gb RAM;
ГА — PENTIUM 133MHz 32Mb RAM;
ПЗ2 — Sun Ultra 2 300MHz 1Gb RAM;
ЛР — Sun Ultra 2 200MHz 512Mb RAM;
МК — PENTIUM III 500MHz;
ПЗ1 — SUN/Sparc 20/801 80MHz + 1Mb SC.
Для алгоритма ИО тип вычислительной машины неизвестен.

Таблица 5
Сравнительная характеристика алгоритмов, 5000 *итераций*

Алгоритм	ε (%), t (с) $n = 30$		ε (%), t (с) $n = 60$		ε (%), t (с) $n = 120$	
ПЗЧО	0,01	0,110	10,69	6,465	31,93	44,673
ЭА	0,13	0,38	10,98	1,14	32,18	14,52
ППО	0,01	0,64	10,94	8,89	33,10	219,86
ГИА	0,02	22,23	10,93	58,03	33,16	318,92
CARA	0,06	1,61	11,46	2,76	34,53	17,00
ГА	0,17	–	11,89	–	35,60	14,05
ПЗ2	–	–	–	–	35,86	109,4
ЛР	–	–	–	–	36,00	72,9
МК	–	–	–	–	36,65	–
ИО	0,23	–	11,90	–	37,68	–
ПЗ1	0,44	–	37,68	–	–	–

Результаты численных экспериментов показывают, что разработанный алгоритм не уступает по качеству получаемых решений лучшим известным алгоритмам. Кроме того, для некоторых примеров из библиотеки PSPLib с помощью данного алгоритма были получены приближенные решения с лучшими значениями целевой функции: в классах j609, j6025, j12016 и j12036 по одному примеру, в классах j6013 и j6029 по два примера, в классах j6045, j12056 по три примера. Полученные результаты позволяют сделать вывод об эффективности данного подхода для решения задачи календарного планирования с ограниченными ресурсами.

6. Заключение

Для задачи календарного планирования с ограниченными ресурсами разработан новый алгоритм поиска с запретами и чередованием окрестностей. В алгоритме используется две окрестности, основанные на активных и T-поздних расписаниях. Окрестности строятся с помощью задачи о многомерном рюкзаке. Численные эксперименты показали, что разработанный алгоритм превосходит многие известные алгоритмы по точности получаемых решений. Представляет интерес опробовать в дальнейшем подобный алгоритм в задачах прямоугольной упаковки.

ЛИТЕРАТУРА

1. **Гимади Э. Х., Залюбовский В. В., Севастьянов С. В.** Полиномиальная разрешимость задач календарного планирования с ограниченными ресурсами и директивными сроками. // Дискрет. анализ и исслед. операций. Сер. 2. 2000. Т. 7, № 1. С. 9–34.
2. **Гончаров Е. Н., Кочетов Ю. А.** Поведение вероятностных жадных алгоритмов для многостадийной задачи размещения // Дискрет. анализ и исслед. операций. Сер. 2. 1999. Т. 6, № 1. С. 12–32.
3. **Гончаров Е. Н., Кочетов Ю. А.** Вероятностный поиск с запретами для дискретных задач безусловной оптимизации // Дискрет. анализ и исслед. операций. Сер. 2. 2002. Т. 9, № 2. С. 13–20.
4. **Ahuja R. K., James O. E., Orlin B., Punnen A. P.** A survey of very large-scale neighborhood search techniques // Discrete Appl. Math. 2002. V. 123, N 1–3. P. 75–102.
5. **Alvarez-Valdés R., Tamarit J. M.** Heuristic algorithms for resource-constrained project scheduling: A review and empirical analysis // Advances in project scheduling. Amsterdam: Elsevier, 1989. P. 113–134.
6. **Baar T., Brucker P., Knust S.** Tabu-search algorithms for the resource-constrained project scheduling problem. // Working Paper, Universität Osnabrück, 1997.
7. **Battiti R., Protasi M.** Reactive local search for the maximum clique problem // Algorithmica. 2001. V. 29, N 4. P. 610–637.
8. **Błażewich J., Lenstra J. K., Rinnooy Kan A. H. G.** Scheduling subject to resource constraints: Classification and complexity // Discrete Appl. Math. 1983. V. 5, N 1. P. 11–24.
9. **Boctor F.** Some efficient multi-heuristic procedures for resource-constrained project scheduling // European J. Oper. Res. 1990. V. 49, N 1. P. 3–13.
10. **Bouleimen K., Lecocq H.** A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem // Techn. Rep. Université de Liège, 1998.

11. Brucker P., Knust S., Schoo A., Thiele O. A branch and bound algorithm for the resource-constrained project scheduling problem // European J. Oper. Res. 1998. V. 107, N 2. P. 272–288.

12. Elmaghraby S. Activity networks: Project planning and control by network models. New York: John Wiley, 1977.

13. Fleszar K., Hindi K. Solving the resource-constrained project scheduling problem by a variable neighborhood search // European J. Oper. Res.(to appear).

14. Glover F., Laguna M. Tabu search. Boston: Kluwer Acad. Publ., 1997.

15. Hansen P., Mladenović N. Developments of variable neighborhood search // Essays and surveys of metaheuristics. Boston: Kluwer Acad. Publ., 2002. P. 415–440.

16. Hartmann S. A competitive genetic algorithm for resource-constrained project scheduling // Naval Res. Logist. 1998. V. 45, N 7. P. 733–750.

17. Hartmann S. Self-adapting genetic algorithm with an application to project scheduling // Techn. Rep. University of Kiel, 1999.

18. Kolisch R. Efficient priority rules for the resource-constrained project scheduling problem // J. Oper. Management. 1996. V. 14, N 3. P. 179–192.

19. Kolisch R. Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation // European J. Oper. Res. 1996. V. 90, N 2. P. 320–333.

20. Kolisch R., Drexl A. Adaptive search for solving hard project scheduling problems // Naval Res. Logist. 1996. V. 43, N 1. P. 23–40.

21. Kolisch R., Schwindt C., Sprecher A. Benchmark instances for project scheduling problems // Project scheduling. recent models, algorithms and applications. Boston: Kluwer Acad. Publ., 1999. P. 197–212.

22. Martello S., Toth P. Knapsack problems. Algorithms and computer implementations. Chichester: John Wiley & Sons, 1990.

23. Merkle D., Middendorf M., Schmeck H. Ant colony optimization for resource-constrained project scheduling // Proc. of the Genetic and Evolutionary Computation Conference, 2000. P. 893–900.

24. Mingozzi A., Maniezzo V., Ricciardelli S., Bianco L. An exact algorithm for resource-constrained project scheduling problem based on a new mathematical formulation // Management Sci. 1998. V. 44, N 5. P. 714–729.

25. Möhring R. H., Schulz A. S., Stork F., Uetz M. Solving project scheduling problems by minimum cut computations // Manag. Sci. 2003. V. 49, N 3. P. 330–350.

26. Nonobe K., Ibaraki T. Formulation and tabu search algorithm for the resource constrained project scheduling problem (RCPSP) // Techn. Rep. 99010. University of Kioto, 1999.

27. Özdamar L., Ulusoy G. An iterative local constraint based analysis for solving the resource-constrained project scheduling problem // J. Oper. Management. 1996. V. 14, N 3. P. 193–208.

28. Palpant M., Artigues Ch., Michelon Ph. Conception d'une métaheuristique et application au problème d'ordonnancement de project à moyens limités // LIA Techn. Rep. 252. Université d'Avignon, 2001.

29. Ribeiro C., Hansen P. Essays and surveys of metaheuristics. Boston: Kluwer Acad. Publ., 2002.

30. Sampson S. E., Weiss E. N. Local search techniques for the generalized resource constrained project scheduling problem // Naval Res. Logist. 1993. V. 40, N 5. P. 665–675.

31. Thomas P., Salhi S. An investigation into the relationship of heuristic performance with network-resource characteristics // J. Oper. Res. Society. 1997. V. 48, N 1. P. 34–43.

32. Thomas P., Salhi S. A tabu search approach for the resource constrained project scheduling problem // J. Heurist. 1998. V. 4, N 2. P. 123–139.

33. Valls V., Ballestín F., Quintanilla S. Resource-constrained project scheduling: a critical activity reordering heuristic // Proc. of the 7th Intern. Workshop on Project Management and Scheduling. Extended Abstracts (Germany, April 17–19, 2000). 2000. P. 282–283.

34. Valls V., Ballestín F., Quintanilla S. A population-based approach to the resource-constrained project scheduling problem // Techn. Rep. 10-2001. University of Valencia, 2001.

35. Węglarz J. Project scheduling. Recent models, algorithms and applications. Boston: Kluwer Acad. Publ., 1999.

36. Yannakakis M. Computational complexity // Local search in combinatorial optimization. Chichester: John Wiley & Sons, 1997. P. 19–55.

Адрес авторов:

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

Статья поступила
17 сентября 2003 г.,
переработанный вариант —
10 ноября 2003 г.