ДИСКРЕТНЫЙ АНАЛИЗ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ
Январь—июнь 2005. Серия 2. Том 12, № 1, 12–36

УДК 519.85

НОВЫЕ ЖАДНЫЕ ЭВРИСТИКИ ДЛЯ ЗАДАЧИ КАЛЕНДАРНОГО ПЛАНИРОВАНИЯ С ОГРАНИЧЕННЫМИ РЕСУРСАМИ*)

Ю. А. Кочетов, А. А. Столяр

Для задачи календарного планирования с ограниченными ресурсами разработаны три вероятностных жадных алгоритма, в которых по-разному трактуется понятие "жадный". Согласно первому алгоритму вычисляются временны́е задержки работ относительно наиболее поздних времен старта в задаче без ресурсных ограничений. Эти задержки используются для ранжирования работ и нахождения приближенного решения по схеме параллельного составления расписаний. Второй алгоритм использует оптимальное решение вспомогательной задачи на узкое место, в которой наряду с временны́ми задержками явным образом учитываются ресурсные ограничения. Идея третьего алгоритма заключается в максимальном использовании выделяемых ресурсов. Для этих целей применяется вспомогательная задача о многомерном рюкзаке. Все алгоритмы являются рандомизированными и используют методы локальной перестройки полученных жадных решений. Приводятся результаты численных экспериментов и сравнение с ранее разработанными алгоритмами.

Введение

В задаче календарного планирования с ограниченными ресурсами (ЗКПОР) задано множество работ, связанных друг с другом условиями предшествования. Эти условия на множестве работ порождают частичный порядок. Без ограничения общности можно считать, что частичный порядок задается ациклическим ориентированным графом с одним источником и одним стоком. Для каждой работы задана длительность ее выполнения и объемы потребляемых ресурсов. Для источника и стока эти величины равны нулю. Суммарный выделяемый объем каждого ресурса считается известным в любой момент времени. Все ресурсы являются возобновляемыми [1], неиспользованные ресурсы пропадают (на-

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

© 2005 Кочетов Ю. А., Столяр А. А.

пример, рабочее время). Потребление ресурсов каждой работой и выделение всех ресурсов предполагается постоянным по времени. Требуется найти расписание выполнения работ, удовлетворяющее условиям предшествования, ограничениям по ресурсам и имеющее минимальное время выполнения всех работ.

Сформулированная задача является NP-трудной в сильном смысле [7]. Более того [24], для любого $\varepsilon > 0$ существование полиномиального приближенного алгоритма с гарантированной оценкой $n^{1-\varepsilon}$ относительного уклонения от оптимума влечет NP=ZPP. Таким образом, при решении данной задачи наряду с полиномиальными алгоритмами целесообразно применять, например, итерационные методы, позволяющие уменьшать погрешность с ростом числа итераций.

Предлагаемый в статье метод решения задачи является серией независимых испытаний вероятностных жадных алгоритмов с последующим улучшением найденных решений методами локальной перестройки. Наилучшее решение, полученное в ходе таких испытаний, является результатом работы данного метода. Следуя [11], класс таких методов будем называть GRASP (Greedy Randomized Adaptive Search Procedure). Аннотированную библиографию с обзором вариантов и приложений GRASP можно найти в [12]. Основная проблема, возникающая при исследовании таких методов, состоит в разработке жадных алгоритмов и удачной их рандомизации, что дает возможность получать много приближенных решений, которые согласно методам GRASP используются в качестве начальных точек для алгоритмов локального спуска. Такой подход позволяет сконцентрировать внимание только на нахождении локальных оптимумов; при этом сложность алгоритма может оказаться неполиномиальной [33].

В данной статье предлагаются три новых вероятностных жадных алгоритма, которые используются в схеме параллельного составления расписаний [16]. В этой схеме рассматриваются возрастающие моменты времени и для каждого из них определяется множество работ, которые будут выполняться параллельно. Выбор множества работ является важным аспектом этой схемы. Рассматриваются три способа выбора таких множеств, в которых по-разному трактуется смысл понятия "жадный".

Первая эвристика определяет временны́е задержки работ относительно наиболее поздних моментов старта в задаче без ресурсных ограничений. Идея этой эвристики состоит в ранжировании работ по временны́м задержкам. Наибольший приоритет получают работы с бóльшими задержками.

Вторая эвристика также использует наиболее поздние моменты старта работ в задаче без ресурсных ограничений, но в отличие от предыдущей эвристики решается оптимизационная задача на узкое место, в которой ресурсные ограничения учитываются явным образом. Задача на узкое место является полиномиально разрешимой, но оптимальных решений может быть экспоненциально много, и они могут существенно отличаться по числу параллельно выполняемых работ и использованию ресурсов. Поэтому среди оптимальных решений ищется решение с минимальными остатками ресурсов.

Третья эвристика старается использовать имеющиеся ресурсы в максимально полном объеме. Эта идея формулируется в виде задачи о многомерном рюкзаке, которая является NP-трудной [22]. Однако точное решение в данном случае не требуется. Для этого метода важно иметь много разных приближенных решений. Для решения задачи о рюкзаке уже разработаны жадные эвристики. В новой эвристике используется их вероятностный вариант с обобщением на многомерный случай.

Все три эвристики часто находят решения с большим отклонением от оптимума. Для сокращения погрешности предусмотрена процедура улучшения, использующая локальный спуск. Это весьма трудоемкая процедура, так как просмотр окрестности и выбор лучшего решения требует многократного вычисления целевой функции. Рассмотрены три известные окрестности разной мощности:

— окрестность линейной мощности, построенная на критических дугах [6];

— окрестность квадратичной мощности [3];

— окрестность экспоненциальной мощности, в которой просматривается только перспективное подмножество линейной мощности [2].

Применение локального спуска по выбранным окрестностям приводит к заметному сокращению погрешности. Априорная оценка числа шагов локального спуска не является полиномиальной и весьма далека от реально наблюдаемых величин. Во всех экспериментах это число не превосходило 5. Тем не менее локальный спуск требует больших затрат машинного времени и в качестве альтернативы локальному спуску исследовалось поведение алгоритма Пинг-понг [2], который, стремясь сократить длину расписания, последовательно переходит от активных расписаний к T-поздним и обратно. Такая процедура не гарантирует получение локального оптимума. Однако во многих случаях результат работы метода GRASP с алгоритмом Пинг-понг на стадии улучшения оказывается локальным оптимумом для данных окрестностей. Для вероятности этого

события построены доверительные интервалы, которые свидетельствуют о предпочтительности алгоритма Пинг-понг ввиду его меньшей трудоемкости.

Статья организована следующим образом. В первом разделе вводятся обозначения и приводится общая схема метода параллельного составления расписаний. Во втором разделе приводятся вероятностные жадные эвристики. В третьем разделе даются определения окрестностей и алгоритма Пинг-понг. В последнем разделе обсуждаются результаты численных экспериментов и приводится сравнение разработанных алгоритмов с известными.

1. Метод параллельного составления расписаний

Введем следующие обозначения:
$J = \{1, \ldots, n\}$ — множество работ, $j = 1$ — источник, $j = n$ — сток;
P_j — множество непосредственных предшественников работы $j \in J$;
K — множество типов ресурсов;
R_k — объем ресурса типа k, выделяемый в каждый момент времени;
r_{jk} — объем ресурса типа k, необходимый для работы j в каждый момент времени;
p_j — длительность выполнения работы j.
Переменные задачи:
s_j — момент начала работы j;
$c_j = s_j + p_j$ — момент окончания работы j.

Задача состоит в построении расписания $S = \{s_j, j \in J\}$, которое удовлетворяет условиям предшествования, ограничениям по ресурсам и минимизирует время окончания всех работ, т. е. требуется найти

$$\min c_n$$

при ограничениях

$$c_i \leqslant s_j,\ i \in P_j; \qquad \sum_{\{\ell | s_\ell \leqslant t < c_j\}} r_{\ell k} \leqslant R_k,\ k \in K,\ t \geqslant 0; \quad s_j \geqslant 0,\ j \in J.$$

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

В основе большинства эвристических алгоритмов для ЗКПОР лежит идея последовательного вычисления величин $s_j, j \in J$. В зависимости от правил выбора очередной работы различают методы последовательного

и параллельного составления расписаний [16]. В данной статье используется второй метод.

Метод параллельного составления расписаний (МПР) выполняет не более n шагов. На шаге m рассматривается момент времени t_m и три множества:

$J(t_m) = \{j \in J \mid c_j \leqslant t_m\}$ — множество работ, завершенных к моменту времени t_m;

$A(t_m) = \{j \in J \setminus J(t_m) \mid s_j \leqslant t_m < c_j\}$ — множество работ, находящихся в процессе выполнения в момент времени t_m;

$D(t_m) = \{j \in J \setminus (J(t_m) \cup A(t_m)) \mid P_j \subseteq J(t_m),\ r_{jk} \leqslant R_k - \sum_{i \in A(t_m)} r_{ik},\ k \in K\}$ — множество работ, которые могут начаться в момент времени t_m.

Если множество $D(t_m)$ не пусто, то из него выбирается несколько работ, которые начнут выполняться в момент времени t_m. Если же это множество пусто, то находится минимальный момент времени, когда закончится одна из выполняемых работ, и осуществляется переход к следующему шагу. Формально алгоритм выглядит следующим образом.

Алгоритм МПР

1. Положить $m := 0,\ t_m := 0,\ J(t_m) := \varnothing,\ A(t_m) := \{1\},\ s_1 := c_1 := 0$.
2. Пока $|J(t_m) + A(t_m)| < n$ выполнять следующее:
   2.1 Положить $m := m + 1$.
   2.2 Найти $t_m := \min\{c_j \mid j \in A(t_{m-1})\}$.
   2.3 Вычислить $J(t_m),\ A(t_m),\ D(t_m)$.
   2.4 Выбрать такое $D' \subseteq D(t_m)$, что $\sum_{j \in D'} r_{jk} \leqslant R_k - \sum_{j \in A(t_m)} r_{jk}$, $k \in K$.
   2.5 Положить $s_j := t_m,\ c_j := t_m + p_j, j \in D'$.

Выбор подмножества $D' \subseteq D(t_m)$ на шаге 2.4 является наиболее важным моментом. Свобода в выборе искомого множества открывает широкие перспективы для развития метода.

2. Вероятностные жадные стратегии

Рассмотрим три вероятностные жадные стратегии выбора подмножества D': выбор на основе временны́х задержек работ, вычисленных относительно оптимального расписания без учета ресурсных ограничений, выбор с помощью задачи на узкое место и выбор с использованием задачи о многомерном рюкзаке.

2.1. Временны́е задержки

Предположим, что множество D' выбрано и $c_i = t_m + p_i, i \in D'$. Рассмотрим работу $j \in D(t_m) \setminus D'$ и определим наиболее ранний момент ее начала, допустимый по ресурсным ограничениям. Искомый момент t'_j является минимальным среди моментов окончания работ из множества $A(t_m) \cup D'$, когда имеется достаточный объем ресурсов для выполнения работы j, т. е.

$$t'_j = \min \gamma$$

при ограничениях

$$r_{jk} \leqslant R_k - \sum_{i \in A(t_m) \cup D',\ c_i > \gamma} r_{ik},\ k \in K,$$

$$\gamma \in \{c_i \mid i \in A(t_m) \cup D'\}.$$

Пусть $n' = |A(t_m) \cup D'|$. Упорядочив множество $A(t_m) \cup D'$ по неубыванию величин p_j, можно найти значение t'_j с временно́й сложностью $O(n' \log n' + n'|K|)$. Отметим, что t'_j зависит от множества D'.

Пусть $\widetilde{t}_j$ — максимум среди всех величин t'_j, соответствующих всевозможным множествам D'. Возможность вычисления этой величины за полиномиальное время сомнительна. Поэтому рассмотрим оценку $\overline{t}_j \geqslant \widetilde{t}_j$, представляющую собой пессимистический прогноз наиболее раннего момента начала работы $j \in D(t_m) \setminus D'$ для любого D'. Определим $\overline{t}_j$ следующим образом. Предположим, что $c_i = t_m + p_i, i \in D' \setminus \{j\}$, и найдем наиболее ранний момент времени γ, когда работе j хватает имеющихся ресурсов, т. е.

$$\overline{t}_j = \min \gamma$$

при ограничениях

$$r_{jk} \leqslant R_k - \sum_{i \in A(t_m) \cup D(t_m) \setminus \{j\},\ c_i > \gamma} r_{ik},\ k \in K,$$

$$\gamma \in \{c_i \mid i \in A(t_m) \cup D(t_m) \setminus \{j\}\}.$$

Все величины $\overline{t}_j$ могут быть найдены за $O(|K|n' \log n')$ операций.

Пусть LF_j — наиболее поздний момент завершения работы j в задаче без учета ограничений по ресурсам. Тогда величина $\Delta_j = \overline{t}_j - LF_j + p_j$ равна задержке работы $j \in D(t_m)$, а $\max\{\Delta_j \mid j \in D(t_m)\}$ характеризует задержку всего проекта. Таким образом, в первую очередь целесообразно включать в множество D' работы с бо́льшими значениями Δ_j, учитывая

ресурсные ограничения. Одна из вероятностных версий такой жадной стратегии может быть представлена следующим образом. В множестве $D(t_m)$ выбирается случайное подмножество $D(q)$. Каждый элемент из $D(t_m)$ включается в $D(q)$ с вероятностью q независимо от других элементов. В множестве $D(q)$ находится элемент с максимальным значением Δ_j и включается в множество D'. Последовательно добавляя таким способом элементы и проверяя ресурсные ограничения, получим случайное множество D'. Вероятностный алгоритм (обозначенный через A_1)выбора множества D', ориентированный на максимальные временны́е задержки, может быть представлен следующим образом.

Алгоритм A_1

1. Положить $\overline{D} := D(t_m),\ D' := \varnothing, \widetilde{R}_k := R_k - \sum_{i \in A(t_m)} r_{ik},\ k \in K$.

2. Выбрать в $\overline{D}$ случайное подмножество $D(q)$. Если $D(q) = \varnothing$, то добавить в $D(q)$ любой элемент из $\overline{D}$.

3. Найти $i_0 \in D(q)$ с максимальной задержкой $\Delta_{i_0} = \max_{i \in D(q)} \Delta_i$.

4. Положить $D' := D' \bigcup \{i_0\};\ \overline{D} := \overline{D} \setminus \{i_0\},\ \widetilde{R}_k := \widetilde{R}_k - r_{i_0 k},\ k \in K$.

5. Для всех $i \in \overline{D}$: если $r_{ik} > \widetilde{R}_k$ для некоторого $k \in K$, то положить $\overline{D} := \overline{D} \setminus \{i\}$.

6. Если $\overline{D} \neq \varnothing$, то перейти на шаг 2.

Результатом работы алгоритма A_1 является множество D', которое зависит от параметра q. При $q = 1$ получаем детерминированный жадный алгоритм. Близкие по смыслу алгоритмы рассматривались в [15, 18]. Ниже будет показано, что параметр q сильно влияет на результаты работы алгоритма. Даже для малой серии испытаний наилучшее найденное решение при $q < 1$ имеет меньшую погрешность, чем детерминированный аналог.

Заметим, что задержка работы j могла быть оценена и другим способом, например, $\Delta_j = t_m - LF_j + p_j,\ j \in D(t_m)$, или $\Delta_j = t_m - s_j^0, j \in D(t_m)$, где $s_j^0,\ j \in J$, — оптимальное решение задачи без учета ресурсных ограничений. При таком определении Δ_j также вводятся временны́е задержки, но не учитывается взаимовлияние элементов множества $D(t_m)$. Экспериментальные исследования показали, что определение задержек через величины $\overline{t}_j$ хоть и более трудоемко, но приводит к лучшим результатам.

2.2. Задача на узкое место

Предположим, что известны оптимальное решение исходной задачи $s_j^*, j \in J$, и некоторое частичное расписание $s_j, j \in J(t_m) \cup A(t_m)$. Тогда для каждой работы $j \in D(t_m)$ можно вычислить задержку $a_j = t_m - s_j^*$, если работа будет включена в множество $D' \subseteq D(t_m)$, и задержку

$b_j = \overline{t}_j - s_j^* \geqslant a_j$, если работа не будет включена в это множество. При известных задержках выбор множества D' следует проводить так, чтобы максимальная задержка работ из множества $D(t_m)$ была бы минимальной, т. е. требуется найти

$$\min_{x_j}\{\max_{j\in D(t_m)}(a_j x_j + b_j(1-x_j))\}$$

при ограничениях

$$\sum_{j\in D(t_m)} r_{jk}x_j \leqslant R_k - \sum_{i\in A(t_m)} r_{ik},\ k\in K,$$

$$x_j \in \{0,1\},\ j\in D(t_m).$$

Эта задача является задачей на узкое место. Оптимальное значение целевой функции достигается на одном из значений a_j или $b_j, j\in D(t_m)$. Упорядочим множество $D(t_m)$ по невозрастанию значений b_j. Согласно этому порядку будем включать работы в множество D'. Как только очередная работа j нарушает хотя бы одно ресурсное ограничение, или для текущего множества D' справедливо неравенство $b_j \geqslant \max_{i\in D'} a_i$, получаем оптимальное решение задачи. Однако на этом процесс пополнения множества D' не заканчивается, поскольку при найденном оптимальном значении оставшиеся ресурсы могут позволить дальнейшее расширение множества D'. Поэтому просмотр работ продолжается и если очередная работа удовлетворяет ресурсным ограничениям, то она также включается в множество D'. Проверка этого условия требует $O(|K|)$ операций, если последовательно пересчитывать остатки ресурсов. Таким образом, оптимальное решение задачи на узкое место может быть найдено за $O(|D(t_m)|\cdot(|K|+\log_2|D(t_m)|))$ операций.

Положим $a_j = t_m - (LF_j - p_j)$, $\quad b_j = \Delta_j = \overline{t}_j - (LF_j - p_j), j\in D(t_m)$, и пусть $x_j^*,\ j\in D(t_m)$, — оптимальное решение задачи на узкое место. Рассмотрим следующий вероятностный алгоритм (обозначенный через A_2) решения задачи на узкое место, который сначала берет некоторые работы со значениями $x_j^* = 1$, а затем пополняет это множество случайным образом работами с $x_j^* = 0$.

Алгоритм A_2

1. Решить задачу на узкое место.

2. Положить $D' = \varnothing, D_1 = \{j\in D(t_m) \mid x_j^* = 1\}, D_0 = \{j\in D(t_m) \mid x_j^*{=}0\}$.

3. Каждый элемент из множества D_1 включить с вероятностью q в множество D'.

4. Упорядочить работы из множества D_0 по невозрастанию величин $\sum\limits_{k\in K} r_{jk}/R_k$.

5. Согласно данному порядку элементы из D_0 включать с вероятностью q в множество D' при соблюдении ресурсных ограничений.

Алгоритм A_2 отдает предпочтение работам, вошедшим в оптимальное решение задачи на узкое место. В этом смысле он является жадным алгоритмом. Отметим, что в рассматриваемой задаче на узкое место оптимальное решение, как правило, не является единственным. Действительно, по определению $b_j \geqslant a_j, j \in D(t_m)$. Если минимум достигается на некотором значении f^*, то в оптимальном решении $x_j^* = 1$ при $b_j > f^*$ и $x_j^* = 0$ при $b_j = f^*$. Значения остальных переменных уже не влияют на оптимум. Другими словами, оптимальные решения могут сильно отличаться по числу выбранных работ. Из множества оптимальных решений целесообразно выбрать такое решение, которое максимально использовало бы имеющиеся ресурсы. Для этих целей на шаге 5 алгоритма используется множество D_0, из которого выбираются работы с бóльшими требованиями по ресурсам. В следующем алгоритме идея максимального использования ресурсов является доминирующей.

2.3. Задача о многомерном рюкзаке

В [2] при определении окрестности рассматривался вариант построения множества D', основанный на решении следующей вспомогательной задачи о многомерном рюкзаке. Найти

$$\max \sum_{j\in D(t_m)} x_j \sum_{k\in K} \frac{r_{jk}}{R_k},$$

при ограничениях

$$\sum_{j\in D(t_m)} r_{jk}x_j \leqslant R_k - \sum_{j\in A(t_m)} r_{jk},\ k \in K,$$

$$x_j \in \{0,1\}, j \in D(t_m).$$

Основная идея такого способа заключается в максимальном использовании ресурсов, имеющихся в момент времени t_m. Сформулированная задача является NP-трудной. Для ее решения используется вероятностный аналог жадных алгоритмов для решения задачи о камнях (см. [22], гл. 4).

Пусть, как и прежде, множество $D(q)$ является случайным подмножеством множества $D(t_m)$. Выберем в $D(q)$ работу с максимальным суммарным весом $\sum_{k\in K} r_{ik}/R_k$ и включим ее в множество D'. Удалим из

$D(t_m)$ выбранную работу и все работы, которым теперь не хватает ресурсов. Эту процедуру будем повторять до тех пор, пока не удалим все работы из $D(t_m)$. Формально этот вероятностный алгоритм (обозначенный через A_3) для решения задачи о многомерном рюкзаке может быть представлен следующим образом.

Алгоритм A_3

1. Положить $D' := \varnothing, \overline{D} := D(t_m),\ \widetilde{R}_k := R_k - \sum\limits_{i \in A(t_m)} r_{ik},\ k \in K.$

2. Выбрать в $\overline{D}$ случайное подмножество $D(q)$. Если $D(q) = \varnothing$, то включить в $D(q)$ любой элемент из $\overline{D}$.

3. Найти $i_0 \in D(q)$ с максимальным весом

$$\sum_{k \in K} r_{i_0 k}/R_k = \max_{i \in D(q)} \sum_{k \in K} r_{ik}/R_k.$$

4. Положить $D' := D' \bigcup \{i_0\}, \overline{D} := \overline{D} \setminus \{i_0\},\ \widetilde{R}_k := \widetilde{R}_k - r_{i_0 k},\ k \in K.$

5. Для всех $i \in \overline{D}$: если $r_{ik} > \widetilde{R}_k$ для некоторого $k \in K$, то положить $\overline{D} := \overline{D} \setminus \{i\}$.

6. Если $\overline{D} \neq \varnothing$, то перейти к шагу 2.

Временна́я сложность алгоритма не превосходит $O(|D(t_m)|^2|K|)$.

3. Локальные улучшения

Рассмотренные вероятностные жадные алгоритмы дают возможность порождать много приближенных решений. Многие из них могут улучшены методами локальной перестройки. Для расписания S рассмотрим три окрестности $N_1(S),\ N_2(S)$ и $N_3(S)$, которые будут использоваться в стандартном алгоритме локального спуска. Первая окрестность была предложена в [6] и строится по так называемым *критическим дугам*. Она имеет линейную мощность относительно числа работ. Вторая окрестность имеет квадратичную мощность и строится по всем парам работ, которые не связанны условиями предшествования. Третья окрестность строится на основе решения вспомогательной задачи о многомерном рюкзаке и использует идею алгоритма МПР для определения соседних решений. Ее мощность экспоненциальна, однако не все элементы окрестности представляют интерес. Среди них выделяется подмножество расписаний, соответствующих приближенным решениям многомерной задачи о рюкзаке, и просматривается только это подмножество, имеющее линейную мощность.

Окрестность $N_1(S)$ определяется для активного расписания S и соответствующего ему списка работ L [6]. Рассмотрим взвешенный ориентированный граф $G = (V, E)$ с множеством вершин $V = J$ и множеством

дуг E, состоящим из пар (i, j) таких, что $s_i + p_i = s_j$ в расписании S. Вершине j припишем вес p_j. Граф G является сетью с одним источником $j = 1$ и возможно несколькими стоками. По построению все пути из источника в сток $j = n$ имеют одинаковую длину c_n. Они называются критическими путями. Дугу (i, j) называют *критической* [6], если она принадлежит некоторому критическому пути и пара работ (i, j) не связана условиями предшествования. Найдем критический путь с минимальным числом дуг. Идея построения окрестности $N_1(S)$ состоит в том, чтобы изменить список L, сделав хотя бы одну критическую дугу этого пути некритической.

Окрестность $N_1(S)$ определяется с помощью трех операторов.

1) Shift(i, j) — оператор, который определяется для критической дуги (i, j), если элемент i предшествует в списке L элементу j. Оператор предъявляет новый список L_1, перемещая элемент i вместе со всеми его последователями $u_1, \dots, u_m$, стоящими перед j в L, непосредственно за элемент j:

$$L = (\dots, i, \dots, u_1, \dots, u_m, \dots, j, \dots) \Rightarrow L_1 = (\dots, j, i, u_1, \dots, u_m, \dots).$$

2) BShift(i, j) — оператор, который определяется для критической дуги (i, j), если j предшествует i. Оператор предъявляет новый список L_2, перемещая ближайший справа от i элемент u, не связанный с i условиями предшествования, непосредственно перед i:

$$L = (\dots, j, \dots, i, \dots, l, \dots, u, \dots) \Rightarrow L_2 = (\dots, j, \dots, u, i, \dots, l, \dots).$$

3) FShift(i, j) — оператор, аналогичный оператору BShift(i, j). Он предъявляет список L_3, перемещая ближайший слева от j элемент u, не связанный с j условиями предшествования, непосредственно за элемент j:

$$L = (\dots, u, \dots, l, \dots, j, \dots, i, \dots) \Rightarrow L_3 = (\dots, l, \dots, j, u, \dots, i, \dots).$$

Каждый из указанных списков порождает активное расписание. Множество таких расписаний образует окрестность $N_1(S)$ мощности $O(n)$.

Окрестность $N_2(S)$, также как и окрестность $N_1(S)$, определяется для активного расписания S и соответствующего списка L. Для каждой пары (i, j), не связанной условиями предшествования, соседний список L_1 получается из списка L переносом работы i вместе со всеми ее последователями, стоящими перед j в L, непосредственно за работу j. Отличие от оператора Shift(i, j) состоит в том, что пара (i, j) не обязана

соответствовать критической дуге и может не принадлежать E. Окрестность $N_2(S)$ состоит из активных расписаний, получаемых из соседних списков. Она имеет мощность $O(n^2)$.

При определении окрестности $N_3(S)$ используются допустимые решения задачи о многомерном рюкзаке. Эта окрестность успешно применялась в алгоритме поиска с запретами [2]. Для каждой работы j выделяется подмножество работ, выполняемых в текущем расписании S параллельно, непосредственно до или после работы j. Для этих работ с помощью алгоритма A_3 вычисляются новые значения переменных s_i. Затем частичное расписание достраивается до полного без изменения порядка выполнения оставшихся работ. Любое допустимое решение задачи о многомерном рюкзаке порождает соседнее решение для расписания S, что приводит к окрестности экспоненциальной мощности. Для сокращения времени поиска в окрестности выделяется подмножество линейной мощности. Далее это подмножество будем обозначать через $N_3(S)$.

Как уже отмечалось выше, поиск соседнего решения с меньшим значением целевой функции требует значительных затрат машинного времени. В качестве альтернативы рассмотрим процедуру из [18, 21,26], которая последовательно переходит от активных расписаний к T-поздним и обратно до тех пор, пока это приводит к уменьшению целевой функции. В [2] эта процедура названа алгоритмом Пинг–понг.

Пусть S_0 — активное расписание и L_0 — соответствующий ему список. Построим список L_1, упорядочив работы по времени их окончания. К полученному списку применим процедуру построения T-позднего расписания, которая, начиная с момента времени $T(S_0) = c_n(S_0)$, в обратном порядке вычисляет наиболее поздние времена окончания работ, не нарушая ограничений по ресурсам. Полученное расписание обозначим через S_1. Построим список L_2, упорядочив работы по началам их выполнения в S_1. По списку L_2 построим активное расписание S_2. Если расписание S_2 имеет меньшую длину чем S_0, то положим $S_0 = S_2$ и повторим эту процедуру. Критерием остановки алгоритма является совпадение значений целевой функции для расписаний S_0 и S_2. Приведенный алгоритм не гарантирует получение локального оптимума по определенным выше окрестностям. Тем не менее, как мы увидим ниже, с большой вероятностью это событие имеет место для окрестностей N_1, N_3. Для окрестности N_2 число шагов для получения локального оптимума оказывается небольшим.

4. Экспериментальные исследования

Численные эксперименты проводились на примерах из электронной

библиотеки тестовых задач PSPLib [19]. Для ЗКПОР в ней содержатся примеры разной степени сложности как для точных методов, так и для приближенных методов вычисления верхних и нижних оценок оптимума. В библиотеке имеется по 480 примеров для n=30, 60, 90, а также 600 примеров для n=120. Число типов ресурсов во всех примерах равно 4. Примеры разбиты на классы, различающиеся числом условий предшествования, жесткостью ограничений на выделение и потребление ресурсов. В каждом классе содержатся по 10 примеров.

В наиболее трудных классах каждая работа требует ресурсы каждого типа, имеет в среднем одного или двух предшественников, а суммарно выделяемые ресурсы минимальны для существования допустимых решений [20]. Примерами таких классов являются:

j30 13, j30 29, j30 45 для $n = 30$,

j60 13, j60 29, j60 45 для $n = 60$,

j120 16, j120 36, j120 56 для $n = 120$.

На них проводилась основная часть экспериментальных исследований.

4.1. Сравнение жадных стратегий

В первом эксперименте сравниваются стратегии выбора подмножества D' на шаге 2.4 алгоритма МПР. Исследуются три детерминированные стратегии. Первая стратегия основана на сортировке допустимого множества по временны́м задержкам $\Delta_j,\ j \in D(t_m)$. Вторая стратегия использует решение задачи на узкое место. Третья стратегия основана на решении задачи о многомерном рюкзаке. Результаты эксперимента приведены в таблице 1.

Т а б л и ц а 1

Средние относительные погрешности для разных стратегий

Класс	МПР+A_1	МПР+A_2	МПР+A_3
j30 13	10,27	9,51	15,92
j30 29	12,10	9,60	16,46
j30 45	9,82	8,84	14,06
j60 13	13,67	12,85	19,57
j60 29	12,68	13,02	19,12
j60 45	13,37	12,02	18,74
j120 16	10,97	11,26	16,30
j120 36	13,43	10,94	17,45
j120 56	14,72	12,79	16,14

В первой колонке указан идентификатор класса тестовых примеров. Во второй, третьей и четвертой колонках показаны средние относительные

погрешности в процентах для первой (A_1), второй (A_2) и третьей (A_3) стратегий соответственно. В этой и следующей таблицах приведены средние значения из 10 испытаний, по одному испытанию для каждого примера в каждом классе.

Первые две стратегии дают близкие результаты с небольшим перевесом в пользу решения задачи на узкое место. Последняя стратегия существенно им проигрывает. По-видимому, это объясняется тем, что решение задачи о многомерном рюкзаке учитывает только ограничения по ресурсам. Первые же две стратегии используют отношения предшествования работ наряду с ресурсными ограничениями.

Второй эксперимент связан с исследованием влияния процедуры Пинг–понг. Структура таблицы 2 аналогична структуре таблицы 1. В ней приведены средние значения относительной погрешности, среднее и максимальное число шагов этой процедуры. Напомним, что шаг состоит в построении двух расписаний: T-позднего и активного. Использование T-поздних расписаний позволяет заметно сократить относительную погрешность. Интересно отметить, что число шагов достаточно мало. Среднее значение, как правило, заключено между 1 и 2, а максимальное значение не превосходит 5. Для всех классов время работы на персональном компьютере Pentium 1200 Mhz трех вариантов алгоритма примерно одинаково и в среднем составляет порядка 0,01 секунды для примеров с 30 работами и порядка 0,03 и 0,12 секунд для задач размерности 60 и 120 работ соответственно.

Таблица 2

Влияние алгоритма ПП (Пинг–понг)

Класс	МПР+A_1+ПП	МПР+A_2+ПП	МПР+A_3+ПП
j30 13	9,18 / 1,5 (3)	8,70 / 1,4 (2)	11,41 / 2,1 (4)
j30 29	8,95 / 1,7 (2)	7,89 / 1,6 (2)	10,52 / 2,0 (3)
j30 45	7,19 / 1,5 (2)	7,10 / 1,6 (2)	10,10 / 1,8 (2)
j60 13	11,88 / 1,8 (2)	9,52 / 2,4 (4)	11,96 / 2,2 (3)
j60 29	10,94 / 1,9 (3)	11,64 / 1,7 (2)	13,96 / 2,0 (2)
j60 45	11,86 / 2,1 (3)	9,99 / 2,1 (4)	12,59 / 2,6 (4)
j120 16	9,71 / 2,2 (3)	9,46 / 2,3 (3)	11,85 / 2,5 (4)
j120 36	11,65 / 2,3 (3)	9,76 / 2,2 (3)	12,47 / 2,5 (4)
j120 56	12,55 / 2,7 (4)	11,23 / 2,3 (5)	13,25 / 2,3 (4)

4.2. Внесение рандомизации

В следующем эксперименте исследуются рандомизированные варианты данных стратегий. Если в алгоритме МПР на шаге 2.4 осуществ-

лять вероятностный выбор множества D', то можно получить несколько расписаний, а затем выбрать из них наилучшее.

На рисунках 1–3 показано изменение средней относительной погрешности в зависимости от параметра рандомизации q и числа построенных расписаний Z. Величина q менялась в пределах от 0,1 до 0,9. Объем выборки составлял 100 испытаний: 10 испытаний для каждого из 10 примеров в каждом классе. Из рисунков видно, что качественное поведение алгоритма мало меняется при смене стратегии. Однако как и в детерминированном случае использование стратегии A_2 оказывается предпочтительнее.

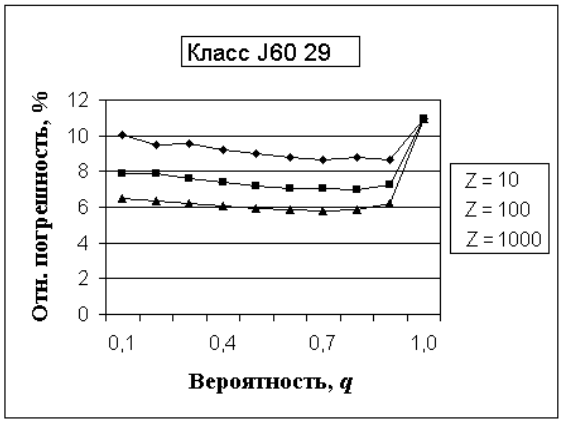

Рис. 1. Влияние рандомизации на погрешность МПР+A_1+ПП

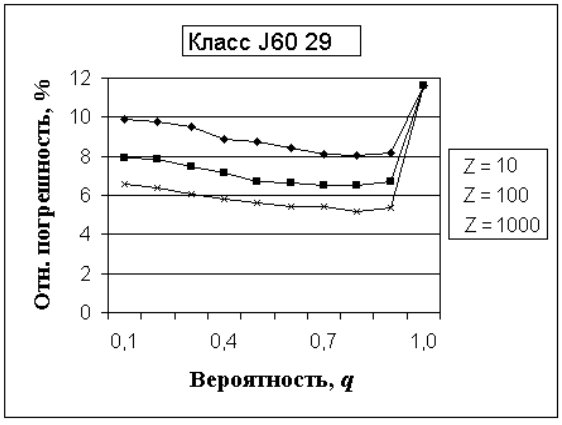

Рис. 2. Влияние рандомизации на погрешность МПР+A_2+ПП

На рисунках представлены результаты для класса j60 29. В остальных классах получаются аналогичные результаты. Они отличаются лишь

абсолютными значениями погрешностей, но общая структура графиков сохраняется.

На рис. 4 показано сравнение рандомизированных стратегий с лучшими значениями параметра q для $Z = 10, 100, 1000$. Оптимальное значение q находится между 0,6 и 0,9. Оно зависит от размерности задачи и убывает с ростом Z. При фиксированном Z увеличение размерности приводит к росту q. Рис. 4 наглядно свидетельствует о преимуществе второй стратегии. Именно она будет использоваться в дальнейших экспериментах.

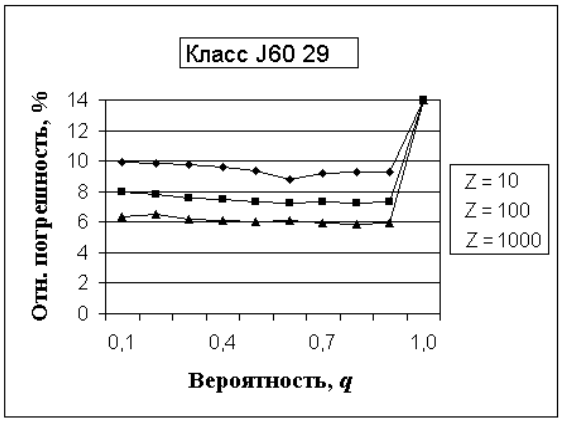

Рис. 3. Влияние рандомизации на погрешность МПР+A_3+ПП

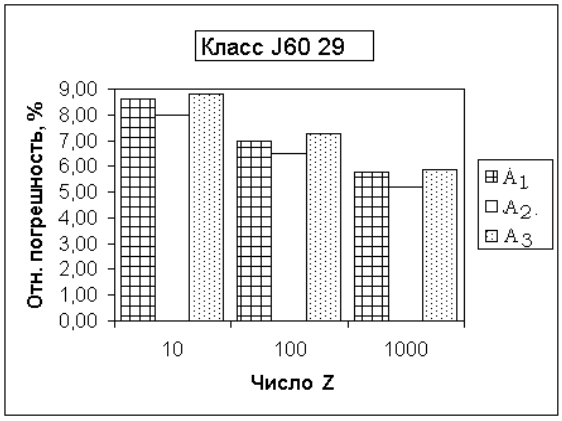

Рис. 4. Сравнение вероятностных стратегий

4.3. Локальный спуск

Следующий эксперимент связан с использованием локального спуска. После работы алгоритма Пинг–понг применяется стандартная процеду-

ра поиска локального оптимума по одной из трех окрестностей, описанных в предыдущем разделе. На рис. 5 показана средняя относительная погрешность локальных оптимумов в зависимости от Z для каждой из окрестностей N_1, N_2, N_3. Эксперимент проводился на задачах из класса j60 29 при $q = 0,7$.

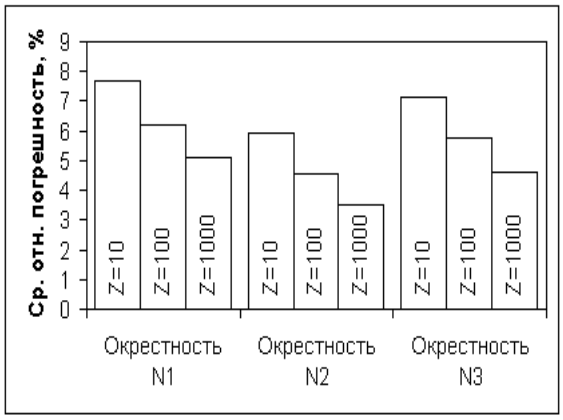

Рис. 5. Относительная погрешность локальных оптимумов

С ростом Z относительная погрешность падает. Вторая окрестность приводит к лучшим локальным оптимумам, что, по-видимому, обусловлено ее квадратичной мощностью. Однако ее просмотр требует больших временны́х затрат. Интересно отметить, что относительная погрешность для второй окрестности почти совпадает с погрешностью для третьей окрестности при 10-ти кратном увеличении Z. Так как просмотр окрестности N_2 требует почти в n раз больше времени, чем просмотр окрестности N_3, то использование третьей окрестности более эффективно несмотря на многократное решение задачи о многомерном рюкзаке. Преимущество третьей окрестности по сравнению с первой представляется очевидным.

В следующем эксперименте проводилось сравнение окрестностей по средней относительной погрешности получаемых локальных оптимумов и числу шагов для завершения локального спуска. Результаты эксперимента приведены в таблице 3. Значение Z выбрано равным 100 и соответствует ста расписаниям, к каждому из которых последовательно применялся алгоритм Пинг–понг и стандартная процедура локального спуска. В качестве жадного алгоритма использовался вероятностный алгоритм A_2 при $0,7 \leqslant q \leqslant 0,9$.

Заметим, что число шагов, которое в таблице 3 обозначено через Depth, в среднем не превосходит единицы для линейных окрестностей

и равно двум или трем шагам для квадратичной окрестности. Другими словами, для окрестностей N_1, N_3 в большинстве случаев результат работы алгоритма Пинг–понг либо является локальным оптимумом, либо локальный оптимум может быть получен за один шаг локального спуска. Для окрестности N_2 требуется чуть больше шагов, но это число также невелико. Картина меняется, если в качестве начальной точки выбрано случайное расписание. В этом случае средняя погрешность получаемых локальных оптимумов и число шагов оказываются почти в два раза больше.

Таблица 3

Погрешность локальных оптимумов и глубина спуска

Класс	ПП ε, %	Окр. N_1 ε, %	N_1 *Depth*	Окр. N_2 ε, %	N_2 *Depth*	Окр. N_3 ε, %	N_3 *Depth*
j30 13	3,69	3,49	0,44	1,23	2,29	2,81	0,56
j30 29	3,31	2,66	0,55	0,88	2,38	2,49	0,82
j30 45	1,95	1,42	0,92	0,18	2,38	1,33	0,70
j60 13	6,25	6,13	0,28	4,81	1,8	5,64	0,88
j60 29	6,55	6,22	0,9	4,56	2,2	5,78	0,90
j60 45	6,33	6,02	0,69	3,84	2,54	5,24	1,07
j120 16	6,57	6,50	0,44	5,47	2,09	6,16	0,99
j120 36	7,31	7,10	0,42	5,62	2,98	6,71	1,38
j120 56	8,40	7,86	0,95	6,43	2,48	7,55	1,31

В таблице 4 приведены доверительные интервалы для вероятности P_0 получения алгоритмом Пинг–понг локального оптимума по каждой из рассматриваемых окрестностей. Результаты расчетов показывают, что для первой и третьей окрестностей получаемые решения довольно часто оказываются локально оптимальными. Для второй окрестности локальные оптимумы получаются редко. Она имеет бóльшую мощность и часто содержит лучшее решение.

Таблица 4

Доверительные интервалы для величины P_0

Класс	N_1	N_2	N_3
j30 13	(0,72 ; 0,77)	(0,12 ; 0,16)	(0,55 ; 0,61)
j60 29	(0,63 ; 0,69)	(0,10 ; 0,14)	(0,40 ; 0,47)
j120 36	(0,60 ; 0,66)	(0,05 ; 0,08)	(0,32 ; 0,38)

В таблице 5 приведены доверительные интервалы для вероятности P_1 получения решения, содержащего локальный оптимум в своей окрестности. Уровень доверия в обоих случаях составлял 0,95. Объем выборки — сто испытаний.

Представленные результаты относились к локальной оптимальности получаемых решений в среднем. Однако вероятностные жадные алгоритмы позволяют находить несколько расписаний ($Z = 10, 100, 1000$) и лучшее из них предъявить в качестве ответа. Интересно знать, с какой вероятностью такое решение оказывается локальным оптимумом. Результаты расчетов представлены в таблице 6.

Таблица 5

Доверительные интервалы для величины P_1

Класс	N_1	N_2	N_3
j30 13	(0,93 ; 0,96)	(0,38 ; 0,44)	(0,84 ; 0,88)
j60 29	(0,88 ; 0,92)	(0,32 ; 0,40)	(0,72 ; 0,78)
j120 36	(0,86 ; 0,90)	(0,19 ; 0,24)	(0,62 ; 0,68)

Таблица 6

Доверительные интервалы для величины P_0 при $Z = 1000$

Класс	N_1	N_2	N_3
j30 13	(0,91; 0,99)	(0,59; 0,77)	(0,88; 0,98)
j60 29	(0,88; 0,98)	(0,47; 0,67)	(0,78; 0,92)
j120 36	(0,78; 0,92)	(0,26; 0,44)	(0,58; 0,76)

Для линейных окрестностей N_1, N_3 данная вероятность достаточно высока и медленно падает с ростом размерности. Для окрестности N_2 она составляет в среднем около 0,5. Локальный спуск по такой окрестности скорее всего позволит сократить погрешность, но потребует вычисления целевой функции во многих точках. Для того чтобы *уравнять в правах* рассматриваемые окрестности, зафиксируем число подсчетов целевой функции, разрешив каждому из алгоритмов обращаться к этой процедуре одинаковое число раз, например, 1000 раз. Тогда для разных окрестностей число Z различно, для линейных окрестностей получим больше локальных оптимумов, сравнение алгоритмов станет более адекватным. Предшествующие расчеты подсказывают, что получаемые решения с высокой вероятностью могут оказаться локальными оптимумами, и усилия по просмотру окрестности будут напрасными. В связи с этим интересно сравнить три варианта алгоритма:

— без локального спуска после работы алгоритма Пинг–понг;

— с одним шагом локального спуска;

— с локальным спуском до получения локального оптимума.

Результаты расчетов приведены в таблице 7. Они свидетельствуют, что

если целевая функция вычисляется не более чем в 1000 точках, то применение методов локального спуска нецелесообразно. Просмотр окрестностей N_1, N_3 и даже N_2 слишком часто оказывается безрезультатным. Лучшее решение легче получить, начав всю процедуру заново, а не пытаться улучшить уже имеющееся решение. Заметим, что ситуация может измениться, если число вычислений целевой функции значительно возрастет. В этом случае дальнейшее сокращение погрешности может потребовать слишком больших значений Z и любые приемы, в том числе и локальный спуск, могут стать оправданными.

Таблица 7

Средняя относительная погрешность для трех вариантов алгоритма

Класс	ПП	ПП+ЛС(1)			ПП+ЛС		
		N_1	N_2	N_3	N_1	N_2	N_3
j30 13	3,13	3,92	5,98	3,87	3,92	4,84	3,90
j60 29	5,84	6,67	9,51	7,01	6,64	9,75	7,25
j120 36	6,80	7,75	9,72	8,25	7,73	9,79	8,51

Таблица 8

Сравнение алгоритмов, $n = 30$

Алгоритм	1000	5000	50000
Жадный алгоритм [30]	0,25	0,13	0,05
Жадный алгоритм [28]	0,30	0,16	0,07
Жадный алгоритм [29]	0,30	0,17	0,09
МПР+A_2+ПП	0,45	0,35	0,25
Жадный алгоритм [32]	0,46	0,28	0,11
Жадный адаптивный алгоритм [27]	0,65	0,44	—
Жадный адаптивный алгоритм [17]	0,74	0,52	—
Эволюционный алгоритм [14]	0,10	0,04	0,00
Поиск с запретами [2]	0,23	0,07	0,01
Рассеивающий поиск [10]	0,27	0,11	0,01
Генетический алгоритм [5]	0,25	0,06	—
Генетический алгоритм [31]	0,27	0,06	0,02
Генетический алгоритм [4]	0,33	0,12	—
Генетический алгоритм [13]	0,38	0,22	0,08
Имитация отжига [8]	0,38	0,23	—
Поиск с запретами [25]	0,46	0,16	0,05
Генетический алгоритм [9]	0,74	0,33	0,16
Поиск с запретами [6]	0,86	0,44	—

4.4. Сравнение с другими алгоритмами

Последний эксперимент связан со сравнением приближенных алгоритмов для решения ЗКПОР. Результаты сравнения приведены в таблицах 8—10. Следуя [18], число вычислений целевой функции ограничивалось величинами 1000, 5000 и 50000. Средняя относительная погрешность приводится для алгоритма МПР, в котором используется процедура A_2 с последующим улучшением алгоритмом Пинг–понг.

Т а б л и ц а 9

Сравнение алгоритмов, $n = 60$

Алгоритм	1000	5000	50000
Жадный алгоритм [30]	11,88	11,62	11,36
МПР+A_2+ПП	12,10	11,98	11,84
Жадный алгоритм [29]	12,14	11,82	11,47
Жадный алгоритм [28]	12,18	11,87	11,54
Жадный алгоритм [32]	12,73	12,35	11,94
Жадный адаптивный алгоритм [27]	12,94	12,58	—
Жадный адаптивный алгоритм [17]	13,51	13,06	—
Рассеивающий поиск [10]	11,73	11,10	10,71
Генетический алгоритм [31]	11,56	11,10	10,73
Эволюционный алгоритм [14]	11,71	11,17	10,74
Поиск с запретами [2]	12,01	11,30	10,77
Генетический алгоритм [5]	11,89	11,19	—
Генетический алгоритм [13]	12,21	11,70	11,21
Генетический алгоритм [4]	12,57	11,86	—
Имитация отжига [8]	12,75	11,90	—
Поиск с запретами [25]	12,97	12,18	11,58
Генетический алгоритм [9]	13,28	12,63	11,94
Поиск с запретами [6]	13,80	13,48	—

Результаты сравнения показывают, что в задачах малой размерности разработанный алгоритм хотя и уступает некоторым известным алгоритмам из аналогичного класса, но с увеличением размерности задачи его выигрыш растет и в задачах с 120 работами он проигрывает только одному из жадных алгоритмов [30].

Следует отметить, что такие метаэвристики, как генетические алгоритмы, поиск с запретами, имитация отжига и т. п., не всегда дают меньшую погрешность, чем жадные алгоритмы. В основном это происходит в тех случаях, когда критерий остановки подразумевает вычисление целевой функции во многих точках. Если же это число предполагается небольшим, как правило, жадные алгоритмы предпочтительнее. В этом смысле удачный выбор алгоритма для решения конкретной задачи обусловлен временны́м фактором.

Заключение

Для задачи календарного планирования с ограниченными ресурсами предложены три новых жадных алгоритма. Во всех алгоритмах применяется известная схема параллельного составления расписаний с последующим улучшением методами локальной перестройки. Численные эксперименты на тестовых примерах из электронной библиотеки PSPLib показали, что предложенные эвристики являются конкурентоспособными при решении задач большой размерности. Они оказываются предпочтительнее методов локального поиска, если целевая функция вычисляется в небольшом числе точек. При большой размерности задачи просмотр окрестности становится трудоемким. Вычисление целевой функции в таком числе точек может оказаться недостаточным и результаты будут посредственными. В этом смысле преимущество вероятностных жадных стратегий будет возрастать с ростом размерности задачи.

Т а б л и ц а 10

Сравнение алгоритмов, $n = 120$

Алгоритм	1000	5000	50000
Жадный алгоритм [30]	35,01	34,41	33,71
МПР+A_2+ПП	35,16	34,98	34,72
Жадный алгоритм [29]	36,24	35,56	34,77
Жадный алгоритм [28]	36,49	35,81	35,01
Жадный алгоритм [32]	38,21	37,47	36,46
Жадный адаптивный алгоритм [27]	39,85	38,70	—
Жадный адаптивный алгоритм [17]	41,37	40,45	—
Генетический алгоритм [10]	34,07	32,54	31,24
Рассеивающий поиск [31]	35,22	33,10	31,57
Эволюционный алгоритм [14]	34,74	33,36	32,06
Поиск с запретами [2]	37,32	34,77	32,61
Генетический алгоритм [5]	36,53	33,91	—
Генетический алгоритм [13]	37,19	35,39	33,21
Муравьиные колонии [4]	—	35,43	—
Генетический алгоритм [8]	39,36	36,57	—
Генетический алгоритм [25]	39,97	38,41	36,44
Поиск с запретами [9]	40,86	37,88	35,85

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. **Кочетов Ю. А., Столяр А. А.** Использование чередующихся окрестностей для приближенного решения задачи календарного планирования с ограниченными ресурсами // Дискрет. анализ и исслед. операций. Сер. 2. 2003. Т. 10, № 2. С. 29–55.

3. **Столяр А. А.** Задача календарного планирования с ограниченными ресурсами: исследование окрестностей для локального поиска // Труды XII Байкальской международной конференции. Т. 6. Иркутск: Изд-во ИрГУ, 2001. С. 46–50.

4. **Alcaraz J., Maroto C.** A robust genetic algorithm for resource allocation in project scheduling // Ann. Oper. Res. 2001. V. 102. P. 83–109.

5. **Alcaraz J., Maroto C., Ruiz R.** Improving the performance of genetic algorithms for the RCPS problem // Proc. of the Ninth International Workshop on Project Management and Scheduling, Nancy, 2004. P. 40–43.

6. **Baar T., Brucker P., Knust S.** Tabu search algorithms and lower bounds for the resource-constrained project scheduling problem // Meta-heuristics. Advances and trends in local search paradigms for optimization. Dordrecht: Kluwer Acad. Publ., 1999. P. 1–18.

7. **Błażewicz 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.

8. **Bouleimen K., Lecocq H.** A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple modes version // Eur. J. Oper. Res. 2003. V. 149, N 2. P. 268–281.

9. **Coelho J., Tavares L.** Comparative analysis of meta-heuristics for the the resource constrained project scheduling problem // Techn. Rep. Department of Civil Engineering, Instituto Superior Tecnico, Portugal. 2003.

10. **Debels D., De Reyck B., Leus R., Vanhoucke M.** A hybrid scatter search/electromagnetism meta-heuristic for project scheduling // To appear in Eur. J. Oper. Res.

11. **Feo T. A., Resende M. G. C.** Greedy randomized adaptive search procedures // J. Global Optim. 1995. V. 6. P. 109–133.

12. **Festa P., Resende M. G. C.** GRASP: An annotated bibliography // Essays and surveys in metaheuristics. Boston: Kluwer Acad. Publ., 2002. P. 325–368.

13. **Hartmann S.** Self-adapting genetic algorithm for project scheduling uder resource constraints // Naval Res. Logist. 2002. V. 49. P. 433–448.

14. **Kochetov Yu., Stolyar A.** Evolutionary local search with variable neighborhood for the resource constrained project scheduling problem // Proc. of 3th Intern. Workshop of Computer Science and Information Technologies, Russia, 2003. P. 96–99.

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

16. **Kolisch R.** Serial and parallel resource-constrained project scheduling methods revisited: theory and computation // Eur. J. Oper. Res. 1996. V. 90, N 2. P. 320–333.
17. **Kolisch R., Drexl A.** Adaptive search for solving hard project scheduling problems // Naval Res. Logist. 1996. V. 43, N 1. P. 23–40.
18. **Kolisch R., Hartmann S.** Heuristic algorithms for the resource-constrained project scheduling problem: classification and computational analysis // Project scheduling: recent models, algorithms and applications. Boston: Kluwer Acad. Publ., 1999. P. 147–178.
19. **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.
20. **Kolisch R., Sprecher A., Drexl A.** Characterization and generation of a general class of resource-constrained project scheduling problems // Management Sci. 1995. V. 41. P. 1693–1703.
21. **Li R.-Y., Willis J.** An iterative scheduling technique for resource-constrained project scheduling // Eur. J. Oper. Res. 1992. V. 56, N 3. P. 370–379.
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 // IEEE Trans. on Evolutionary Computation. 2002. V. 6, N 4. P. 333–346.
24. **Möhring R. H., Schulz A. S., Stork F., Uetz M.** Solving project scheduling problems by minimum cut computations // Management Sci. 2003. V. 49, N 3. P. 330–350.
25. **Nonobe K., Ibaraki T.** Formulation and tabu search algorithm for the resource constrained project scheduling problem // Essays and surveys in metaheuristics. Boston: Kluwer Acad. Publ., 2002. C. 557–588.
26. **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.
27. **Schirmer, A.** Case-based reasoning and improved adaptive search for project scheduling // Naval Res. Logist. 2000. V. 47, N 3. P. 201–222.
28. **Tormos P., Lova A.** A competitive heuristic solution techniques for resource-consrtained project scheduling // Ann. Oper. Res. 2001. V. 102. P. 65–81.
29. **Tormos P., Lova A.** An efficient multi-pass heuristic for project scheduling with consrtained resourcs // Internat. J. Production Research. 2003. V. 41, N 5. P. 1071–1086.
30. **Tormos P., Lova A.** Integrating heuristics for resource-consrtained project scheduling: One step forward // Techn. Rep. Department of Statistics and Operations Research, Universidad Politecnica de Valencia, 2003.

31. Valls V., Ballestin F., Quintanilla S. A hybrid genetic algorithm for the RCPSP // Techn. Rep. Departament of Statistics and Operation Research, University of Valencia, 2003.

32. Valls V., Ballestin F., Quintanilla S. Justification and RCPSP: A technique that pays // Eur. J. Oper. Res. 2005. V. 165, N 2. P. 375–386.

33. 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.
E-mail: jkochet@math.nsc.ru,
asto@math.nsc.ru

Статья поступила
15 июня 2004 г.

Переработанный вариант —
25 апреля 2005 г.