E. N. Goncharov, A stochastic greedy algorithm for the resource-constrained project scheduling problem, *Diskretn. Anal. Issled. Oper.*, 2014, Volume 21, Number 3, 11–24

ДИСКРЕТНЫЙ АНАЛИЗ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ
Май—июнь 2014. Том 21, № 3. С. 11–24

УДК 519.8

СТОХАСТИЧЕСКИЙ ЖАДНЫЙ АЛГОРИТМ ДЛЯ ЗАДАЧИ КАЛЕНДАРНОГО ПЛАНИРОВАНИЯ С ОГРАНИЧЕННЫМИ РЕСУРСАМИ [*)]

Е. Н. Гончаров

Аннотация. Рассматривается многономенклатурная одномодальная задача календарного сетевого планирования в условиях ограниченных ресурсов по критерию минимизации срока выполнения проекта. Ресурсы предполагаются возобновимыми (нескладируемыми). Предлагается быстрый стохастический жадный алгоритм. Качество алгоритма исследовано в серии вычислительных экспериментов, тестовые примеры для которых взяты из библиотеки тестовых задач PSPLIB. Среди жадных алгоритмов предложенный алгоритм занимает одни из лучших позиций, а на тестовых примерах J60 из PSPLIB по 50000 испытаний он показал лучший результат. Среди всех алгоритмов он оказался конкурентоспособным, уступив лишь генетическим алгоритмам и комбинированным на их основе.

Ключевые слова: задача календарного планирования, ограниченный ресурс, нескладируемый ресурс, эвристический алгоритм.

Введение

В задаче календарного планирования с ограниченными ресурсами (ЗКПОР) требуется найти расписание выполнения работ с минимальным сроком завершения проекта. При этом учитываются технологические ограничения предшествования работ и не допускаются их прерывания.

Частичный порядок на множестве работ задаётся ациклическим ориентированным графом. Для каждой работы известны длительность её выполнения, множество потребляемых ею ресурсов, их объём и профиль потребления ресурсов на протяжении выполнения работы.

[*)] Исследование выполнено при финансовой поддержке Российского фонда фундаментальных исследований (проект № 13–07–00809), целевой программы № 2 Президиума РАН (проект № 227), а также целевой программы СО РАН (интеграционный проект № 7Б).

Считаем известным объём выделяемого ресурса в каждый момент времени периода планирования с ограниченными ресурсами и полагаем эти объёмы неограниченными за его пределами. Все ресурсы являются возобновимыми.

По классификации из [9] сформулированная задача обозначается через $PS|prec|C_{\max}$. Она является обобщением известной задачи job-shop scheduling и принадлежит классу NP-трудных задач [7].

Рассматриваемая задача имеет широкое применение на практике и привлекает внимание многих исследователей. В силу труднорешаемости поставленной задачи целесообразно построение малотрудоёмких приближённых алгоритмов. Большой обзор разработанных для ЗКПОР эвристических алгоритмов содержится в [17, 21, 22].

В отечественной и зарубежной литературе применяется разная классификация ограниченных ресурсов [3, 13]. Если в зарубежной литературе они традиционно подразделяются на возобновимые и невозобновимые ресурсы (см., например, [9]), то в российской — на складируемые и нескладируемые. Ресурс назовём *складируемым*, если он, будучи неистраченным в момент t, может быть использован в любой момент $t' > t$, в противном случае — *нескладируемым*. Как отмечено в [3], понятие нескладируемого ресурса в точности совпадает с понятием возобновимого ресурса.

Задача календарного планирования со складируемыми ограниченными ресурсами отличается от задачи с нескладируемыми ресурсами, для этой задачи известны эффективные методы решения. В [1, 13] для решения ЗКПОР с директивными сроками и ограничениями на складируемые ресурсы предложен асимптотически точный алгоритм, время работы которого зависит от числа работ N как функция порядка $N \log N$, а погрешность стремится к нулю с ростом размерности задачи. В [3] показано, что при отсутствии ограничений на нескладируемые ресурсы и с целочисленными длительностями работ нахождение точного решения сформулированной задачи возможно за полиномиальное время.

В настоящей работе рассматривается задача с ограниченными нескладируемыми ресурсами в случае, когда отсутствуют директивные сроки. Для этой задачи предлагается стохастический эвристический алгоритм. Он представляет собой серию независимых испытаний вероятностного жадного алгоритма, в качестве решения берётся наилучшее из полученных решений. Использовалась также модификация этого алгоритма, в которой для получаемых решений применялась процедура их локального улучшения.

Для этой цели используем предложенный в [14] детерминированный жадный алгоритм, временна́я сложность которого зависит от числа работ N как $N \log N$. Основная идея этого алгоритма состоит в том, что наряду с задачей с нескладируемыми ресурсами рассматриваем релаксированную задачу, в которой все ресурсы складируемы. Для решения такой задачи используется приближённый алгоритм [1, 13]. В общем случае для исходной задачи с нескладируемыми ресурсами полученное решение недопустимо, будем использовать его как отправную точку для построения приближённого решения исходной задачи.

В электронной библиотеке тестовых задач [23] представлено большое количество примеров ЗКПОР разной степени сложности. Было проведено тестирование качества получаемых решений на сериях примеров J60 и J120 из этой библиотеки, результаты численных экспериментов приводятся. Получаемые решения имели сравнительно небольшие отклонения от наилучших найденных значений. Приводится сравнение данного алгоритма с лучшими алгоритмами других авторов как в классе жадных, так и в классе произвольных алгоритмов. Среди жадных алгоритмов предложенный алгоритм занимает одни из лучших позиций, а на тестовых примерах J60 из PSPLIB по 50000 испытаний он показал лучший результат. Среди всех алгоритмов он оказался конкурентоспособным, уступив лишь генетическим алгоритмам и комбинированным на их основе.

1. Математическая модель

Будем рассматривать проект как ориентированный ациклический граф $G = (V, U)$ (так называемый граф *редукции*, или *частичного порядка*), где V — множество вершин-событий сетевой модели, $U \subset V \times V$ — множество дуг-работ, $|U| = N,\ |V| = m$. Множество U состоит, во-первых, из работ, связанных с потреблением ресурсов и называемых *фактическими*. Их количество обозначим через n. Помимо фактических это множество содержит *фиктивные* работы, не потребляющие никакого ресурса и служащие для задания частичного порядка на множестве фактических работ. Число фиктивных работ зависит от заданного частичного порядка на множестве фактических работ и оценивается величиной $n^2/2$.

Обозначим через $\alpha(j)$ начальное событие, через $\beta(j)$ — концевое событие работы $j \in U$. Для каждой работы j будем считать известным множество $\mathrm{Pred}(j)$ её непосредственных предшественников.

Пусть M — множество типов ресурсов, задействованных в проекте, p_j — длительность работы $j \in U$, T_k — длительность интервала планирования (горизонт планирования) с ограничением на ресурсы типа $k \in M$

(при $t > T_k$ предполагается, что ограничение на ресурс типа k не накладывается), R_t^k — ограничения на объём расходования ресурса типа $k \in M$ в интервал времени $[1, T_k]$.

Каждая фактическая работа может потреблять произвольное количество ресурсов. Для каждой работы $j \in U$ и ресурса $k \in M$ задаётся объём потребления $r_{jk}(t)$ ресурса типа k работой j в момент t от начала её выполнения. Предполагаем, что $r_{jk}(t) = 0$ при $t \notin [0, p_j]$.

Совокупность S моментов $\{s_j\}$, $j \in U$, начала выполнения работ называется *допустимым расписанием*, если соблюдаются ограничения, обусловленные отношением предшествования, и ресурсные ограничения.

Пусть $U(t) = \{j \mid s_j < t \leqslant s_j + \tau_j\}$ — множество работ, выполняемых в единичном интервале $[t-1, t)$ при расписании S. Прерывания работ не разрешаются. Задача заключается в нахождении допустимого расписания $S = \{s_j\}$ с минимальным временем завершения проекта $C_{\max}(S)$.

Рассматриваемую задачу в этом случае можно формализовать следующим образом: минимизировать время завершения проекта

$$C_{\max}(S) = \max_{j \in U}(s_j + p_j) \ \longrightarrow \min_{s_j} \tag{1}$$

при условиях

$$s_i + p_i \leqslant s_j, \quad i \in \mathrm{Pred}(j),\ j \in U, \tag{2}$$

$$\sum_{j \in U(t)} r_{jk}(t - s_j) \leqslant R_t^k, \quad k \in M,\ t = 1, \ldots, T_k, \tag{3}$$

$$s_j \in \mathbb{Z}^+, \quad j \in J. \tag{4}$$

Неравенства (2) — ограничения предшествования работ. Соотношение (3) обеспечивает соблюдение ограничений по нескладируемым ресурсам: суммарное количество ресурса типа k, потребляемое всеми работами, выполняемыми в каждый момент времени t, не должно превышать имеющегося в наличии количества этого ресурса в данный момент времени. Наконец, (4) определяет переменные выбора.

2. Задача со складируемыми ресурсами

Основная идея алгоритма решения исходной задачи состоит в том, что сначала решается релаксированная задача, где все ограниченные ресурсы объявляются складируемыми. Полученное решение релаксированой задачи затем используется для построения приближённого решения исходной задачи.

Рассмотрим наряду с ограничением (3) для нескладируемых ресурсов следующее ограничение ЗКПОР для складируемых ресурсов:

$$\sum_{t'=1}^{t} \sum_{j \in U(t')} r_{jk}(t' - s_j) \leqslant \sum_{t'=1}^{t} R_{t'}^{k}, \quad k \in M,\ t = 1, \ldots, T_k. \tag{5}$$

Соотношения (5) обеспечивают соблюдение ресурсных ограничений по складируемым ресурсам, а задача $(1), (2), (5), (4)$ является задачей календарного планирования со складируемыми ресурсами.

Отметим, что ЗКПОР с ограничениями на ресурсы складируемого типа (5) радикально отличается от задачи с ограничениями на ресурсы нескладируемого типа (3). Если все ресурсы исключительно складируемого типа, то для довольно широкого класса задач (с целочисленными длительностями работ) задача является полиномиально разрешимой [3], в то время как задача с ограничениями на ресурсы нескладируемого типа NP-трудна [7].

Помимо упомянутого полиномиального алгоритма [3] известен приближённый асимптотически точный алгоритм для решения ЗКПОР, причём для задачи в более общей постановке. В этой постановке ресурсные ограничения только складируемого типа, моменты начала выполнения работ могут быть неотрицательными вещественными числами, а также могут учитываться директивные сроки выполнения событий. Время работы этого алгоритма зависит от числа работ N как функция порядка $N \log N$, а относительная и абсолютная погрешности стремятся к нулю с ростом размерности задачи [1, 13]. Для решения задачи $(1), (2), (4), (5)$ будем использовать именно этот алгоритм. Обозначим через S множество приближённых решений задачи, полученных данным алгоритмом.

Для допустимых расписаний длительности $C_{\max}(S)$ из S рассмотрим также выполнение следующего дополнительного критерия: динамика потребления ресурсов по возможности меньшим образом отличается от предлагаемой динамики лимитированных ресурсов, т. е. на множестве расписаний длительности $C_{\max}(S)$ достигает минимума функция

$$\sum_{k \in M} \sum_{t=1}^{C_{\max}(S)} \left| R_k^t - \rho_k^t(S') \right| \to \min_{S' \in \{S\}}, \tag{6}$$

где $\rho_k^t(S')$ — объём потреблённых ресурсов типа k в момент t.

Задача (1), (2), (5), (4) с дополнительным критерием (6) также решается приближённо с временно́й сложностью, зависящей от количества работ N как $O(N \log N)$ [1, 13]. Назовём полученное приближённое решение данной задачи *оценочным* и обозначим его через S^{est}.

3. Метод решения задачи

В настоящей работе предлагается эвристический вероятностный алгоритм для решения ЗКПОР. Он использует известный метод, который представляет собой серию из λ независимых испытаний вероятностного жадного алгоритма, в качестве решения берётся наилучшее из полученных решений.

При каждом испытании используется жадный алгоритм, представленный в [14].

3.1. Жадный алгоритм. Согласно классификации типов эвристических алгоритмов для ЗКПОР, приведённой в [22], данный алгоритм можно отнести к так называемой последовательной схеме генерации расписаний (serial schedule generation scheme, или serial SGS).

Коротко суть предлагаемого алгоритма можно изложить следующим образом. Он состоит в точности из N шагов, на каждом из которых выбирается ровно одна работа из числа не рассмотренных и для неё определяется (назначается) время её старта, при этом соблюдаются ограничения предшествования работ и ограничения на ресурсы. Далее будем такую процедуру именовать *наложением работы на календарь*. На каждом шаге $g = 1, \dots, N$ будем считать известными множества S_g работ, уже наложенных на календарь, и D_g — работ, не наложенных на календарь, таких, что все непосредственно предшествующие им работы принадлежат множеству S_g. Руководствуясь некоторым правилом, выбираем работу j из множества D_g и назначаем ей минимальное время начала её исполнения такое, что выполнены ограничения предшествования работ и ресурсные ограничения.

В последовательной схеме генерации расписаний важную роль играет порядок наложения работ на календарь. Правило выбора наиболее приоритетной работы из множества D_g будем определять, исходя из минимального значения функции приоритетов $v(j)$, $j \in D_g$, которую называют также *весовой функцией*. В [22] приведён обзор используемых в литературе функций приоритетов. Они могут использовать, например, структуру сети, времена свершения событий, интенсивность потребления ресурсов и т. п.

В качестве такой функции возьмём один из её вариантов, представленных в [14], показавший для детерминированного алгоритма наилучший результат:

$$v(j) = c_1 s_j - c_2 \max_{t,k} w_k r_{jk}(t), \quad j = 1, \dots, N. \tag{7}$$

Здесь $c_1, c_2 > 0$, $s_j \in S^{\text{est}}$, а $w_k,\ k \in M$, — весовые коэффициенты ресурсов, которые полагаются равными 1, кроме одного, который полагался равным достаточно большому числу. В [22] данный метод построения функции приоритетов именуется Multipriority rule method.

3.2. Правило рандомизации. Обозначим через P, $0 < P < 1$, вероятность выбора работы для наложения на календарь. На каждом шаге жадного алгоритма берём наиболее приоритетную (с наименьшим значением $v(j)$) из нерассмотренных работ множества D_g и с вероятностью P выбираем её для последующего наложения на календарь. В случае неудачи выбора берём следующую по приоритетности работу и повторяем процедуру выбора для неё. В случае, когда процедура выбора оказалась неудачной для всех работ из $j \in D_g$, выбор такой работы производим случайным образом равновероятно.

3.3. Схема алгоритма. Обозначим неиспользованный остаток ресурса типа k в момент времени t через $\widetilde{R}_t^k$. Тогда вероятностный алгоритм, применяемый при каждом независимом испытании, может быть формализован следующим образом.

Алгоритм $\mathcal{A}$

Шаг 1. Решаем задачу (1), (2), (5), (4) с дополнительным критерием (6) и находим оценочное расписание S^{est}.

Шаг 2. Определяем весовую функцию $v(j),\ j = 1, \dots, N$, используя для этого оценочное расписание S^{est}, коэффициенты $c_1, c_2 > 0$ и весовые коэффициенты ресурсов $w_k,\ k \in M$.

Шаг 3. Положим $s_j := 0,\ j = 1, \dots, N,\ te_i := 0,\ i = 1, \dots, m,$ $\widetilde{R}_t^k := R_t^k,\ k \in M,\ t = 1, \dots, T_k$.

Шаг 4. For $g := 1$ to N do

{

Находим D_g.

В соответствии с вероятностным правилом выбора выбираем работу для наложения на календарь. Пусть $j \in D_g$ — выбранная работа.

$s_j := \min\big\{t \geqslant te_{\alpha(j)} \mid r_{jk}(\tau) \leqslant \widetilde{R}_t^k,\ k \in M,\ \tau \in [t, t + p_j]\big\}$,

$S_g := S_{g-1} \cup \{j\}$.

Пересчитываем $\widetilde{R}_t^k,\ k \in M,\ \tau \in [t, t + p_j]$,

$te_{\beta(j)} := \max\{te_{\beta(j)}, s_j + p_j\}$.
}

В результате работы алгоритма получаем приближённое решение исходной задачи — расписание s_j.

3.4. Локальное улучшение решений. При каждом независимом испытании вероятностного алгоритма в ходе выполнения алгоритма $\mathcal{A}$ получаем различные приближённые решения задачи. Можно предпринять попытку улучшить эти решения, применяя к ним алгоритмы локального улучшения решений. Воспользуемся идеей известного Forward-backward improvement (FBI) алгоритма.

Впервые FBI-алгоритм предложен в [24] и рассматривался также в [6, 26]. FBI-алгоритм не является полиномиальным, однако многие исследователи отмечали сравнительно небольшое число его итераций. Данный алгоритм используем каждый раз, когда в ходе работы алгоритма $\mathcal{A}$ будет находиться решение, целевая функция на котором лучше ранее найденных.

4. Вычислительный эксперимент

Качество предложенного алгоритма тестировалось на примерах из библиотеки тестовых задач PSPLIB [23]. Эти примеры являются частным случаем исходной задачи, когда потребление ресурса постоянно, оно происходит одномоментно в начале каждого временно́го периода и длительности работ — целые. Для численного эксперимента были взяты два множества тестовых примеров J60 и J120. Серия J60 состоит из 480 примеров, число работ n в них равно 62 (считая начальную и концевую работы), а в серии J120 600 примеров и $n = 122$.

Как отмечено в разд. 1, общее число работ N в исходном графе G зависит не только от числа работ n, но и от характера их взаимосвязей (частичного порядка) на множестве этих работ для каждого исходного конкретного примера, а сверху эта величина ограничена величиной $O(n^2)$. Таким образом, общая временна́я сложность алгоритма $\mathcal{A}$ зависит от параметра n уже как функция порядка $n^2 \log n$.

Следует заметить, что среди множества работ этой сетевой модели будет значительное количество фиктивных работ (лишь ровно n работ в этом случае будут фактическими), и ещё до начала работы алгоритма эта сетевая модель может быть подвержена оптимизации. Как показал вычислительный эксперимент, в классе примеров J60 среднее число работ в графе G после такой оптимизации составило $N = 115{,}8$. Отметим, что в этом случае $n = 62$ и $N = 1{,}87n$. На множестве примеров J120 $N = 227{,}1$ и $N = 1{,}86n$.

Для весовой функции (7), используемой в ходе работы жадного алгоритма при каждом испытании, применены значения коэффициентов $c_1 = 1$, $c_2 = 0{,}1$. Такие коэффициенты были выбраны в соответствии с результатами, полученными в ходе вычислительных экспериментов приведённых в [14], как наилучшие.

Для формирования весовой функции необходимо определить значение параметра P (вероятности выбора работы из множества D_g). Для выбора значения P был выполнен численный эксперимент. Он проведён на серии тестовых задач J60, для каждого примера из этой серии количество испытаний λ равно 1000, а P принимали значения от 0,2 до 1 с шагом 0,1. Результаты этого эксперимента представлены на рис. 1.

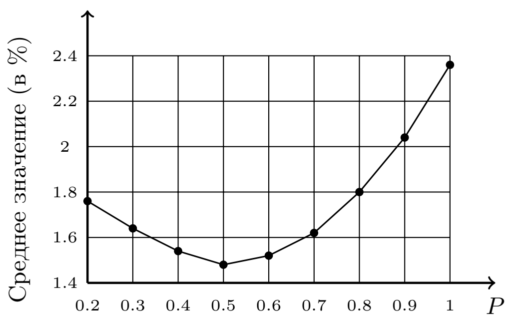

Рис. 1. Средние отклонения от наилучших решений в зависимости от параметра P на серии задач J60, 1000 итераций

Как видно из рисунка, наименьшие отклонения полученных решений наблюдались при $P = 0{,}5$. Отклонения полученных решений вычислялись от наилучших найденных решений по каждой задаче. Далее все численные эксперименты проводились при $P = 0{,}5$.

Следующий численный эксперимент был поставлен для определения качества получаемых решений при использовании алгоритмов $\mathcal{A}$ и $\mathcal{A}$ + FBI. Вычисления производились на сериях примеров J60 и J120 из PSPLIB. Величина λ равна 1000 и 50000. В качестве показателя качества решения использовалась средняя относительная погрешность, вычисляемая по отношению к величине критического пути для каждого примера. Для алгоритма $\mathcal{A}$ + FBI вычисления целевой функции в ходе работы FBI включены в счётчик числа испытаний λ.

В табл. 1 и 2 приведены средние значения отклонений полученных решений в классе жадных алгоритмов (табл. 1) и различных типов алгоритмов (табл. 2) в зависимости от величины критического пути (в процентах) для серии J60 из библиотеки PSPLIB.

В табл. 3 и 4 приведены аналогичные результаты в классе жадных алгоритмов (табл. 3) и различных типов алгоритмов (табл. 4) для серии J120 из этой же библиотеки PSPLIB.

Таблица 1

Сравнение жадных алгоритмов на множестве примеров J60

Алгоритм	Средние отклонения	
	$\lambda = 1000$	$\lambda = 50000$
Алгоритм $\mathcal{A}$ + FBI	11,98	11,35
Tormos, Lova [29]	11,88	11,36
Алгоритм $\mathcal{A}$	12,02	11,36
Кочетов, Столяр [6]	12,10	11,84
Tormos, Lova [30]	12,14	11,47
Tormos, Lova [31]	12,18	11,54
Valls, Ballestin, Quintanilla [32]	12,73	11,94
Schirmer [28]	12,94	
Kolisch, Drexl [20]	13,51	

Таблица 2

Сравнение алгоритмов на множестве примеров J60

Алгоритм	Ссылка	Средние отклонения	
		$\lambda = 1000$	$\lambda = 50000$
GANS	Proon, Jin [27]	11,35	10,52
DBGA	Debels, Vanhoucke [12]	11,31	10,68
GA	Debels, Vanhoucke [12]	11,45	10,68
Scatter search-FBI	Debels et al. [11]	11,73	10,71
GA-Hybrid, FBI	Valls et al. [33]	11,56	10,73
GA,TS-Path re-linking	Кочетов, Столяр [5]	11,71	10,74
Алгоритм $\mathcal{A}$+FBI	Эта статья	11,98	11,35
Алгоритм $\mathcal{A}$	Эта статья	12,02	11,36
GA, FBI	Valls et all [32]	12,21	10,74
GA-Self adapting	Hartmann [15]	12,21	11,21
GA-Activity list	Hartmann [16]	12,68	11,23
Sampling-LFT, FBI	Tormos, Lova [31]	12,81	11,54
SA-Activity list	Bouleimen, Lecocq [8]	12,75	—
TS-Activity list	Nonobe, Ibaraki [25]	12,97	11,58
GA-Late join	Coelho, Tavares [10]	13,28	11,94
GA-Priority rule	Hartmann [16]	13,30	12,26
Sampling-Adaptive	Kolisch, Drexl [20]	13,51	—
Sampling-WCS	Kolisch [18]	13,66	—
Sampling-LFT	Kolisch [19]	13,59	12,83

Из табл. 1 и 3 видно, что разработанный алгоритм занимает ведущие позиции в классе жадных алгоритмов. На множестве примеров J60 он первый, а на J120 занимает лидирующие позиции. Дополнительное применение алгоритма FBI для локального улучшения решений, полученных в результате работы жадного алгоритма, на обоих множествах примеров принесло эффект, и средние отклонения решений улучшены. Впрочем, ценой такого улучшения является утрата полиномиальности алгоритма $\mathcal{A}$+FBI в сравнении с $\mathcal{A}$.

Т а б л и ц а 3

Сравнение алгоритмов на множестве примеров J120

Алгоритм	Средние отклонения	
	$\lambda = 1000$	$\lambda = 50000$
Tormos, Lova [29]	35,01	33,71
Кочетов, Столяр [6]	35,16	34,72
Tormos, Lova [30]	36,24	34,77
Алгоритм $\mathcal{A}$ + FBI	36,43	35,08
Tormos, Lova [31]	36,49	35,01
Алгоритм $\mathcal{A}$	36,77	35,33
Valls, Ballestin, Quintanilla [32]	38,21	36,46
Schirmer [28]	39,85	—
Kolisch, Drexl [20]	41,37	—

Среди всевозможных эвристических алгоритмов (см. табл. 2 и 4) предложенный алгоритм незначительно уступает лишь группе алгоритмов, являющихся либо генетическими, либо гибридными на основе генетических. С учётом малой временно́й сложности данный алгоритм является конкурентоспособным среди них. Отметим также такую особенность предложенного алгоритма: с возрастанием числа λ независимых испытаний среднее отклонение решений уменьшается, но не столь значительно, как у других алгоритмов, представленных в табл. 1–4. Это свидетельствует о том, что он сравнительно быстро находит хорошие решения.

Предложенный алгоритм использовался на конкретных примерах большой размерности, разработанных в ИЭиОПП СО РАН для экономических проектов [1, 4]. Количество работ в этих примерах доходило до 5000, а количество ограниченных ресурсов — до 12, причём среди них были как складируемые, так и нескладируемые. Профили выделения ресурсов и потребления их работами были произвольными. Время счёта в этих примерах составило несколько секунд на настольном персональном компьютере.

Т а б л и ц а 4

Сравнение алгоритмов на множестве примеров J120

Алгоритм	Ссылка	Средние отклонения	
		$\lambda = 1000$	$\lambda = 50000$
GANS	Proon, Jin [27]	33,45	30,45
DBGA	Debels, Vanhoucke [12]	33,55	30,69
GA	Debels, Vanhoucke [12]	34,19	30,82
GA-Hybrid, FBI	Valls et al. [33]	34,07	31,24
Scatter search-FBI	Debels et al. [11]	35,22	31,57
GA, FBI	Valls et al. [32]	35,39	31,58
GA,TS-Path re-linking	Кочетов, Столяр [5]	34,74	32,06
Алгоритм $\mathcal{A}$+FBI	Эта статья	36,43	35,08
Алгоритм $\mathcal{A}$	Эта статья	36,77	35,33
GA-Self adapting	Hartmann [15]	37,19	33,21
GA-Activity list	Hartmann [16]	39,37	34,04
Sampling-LFT, FBI	Tormos, Lova [31]	36,49	35,01
TS-Activity list	Nonobe, Ibaraki [25]	40,86	35,85
GA-Late join	Coelho, Tavares [10]	39,97	36,44
SA-Activity list	Bouleimen, Lecocq [8]	42,81	—
GA-Priority rule	Hartmann [16]	39,93	36,51
Sampling-LFT	Kolisch [19]	39,6	37,74
Sampling-WCS	Kolisch [18]	39,65	—
Sampling-Adaptive	Kolisch, Drexl [20]	41,37	—

ЛИТЕРАТУРА

1. **Гимади Э. Х.** О некоторых математических моделях и методах планирования крупномасштабных проектов // Модели и методы оптимизации. Т. 10. — Новосибирск: Наука, 1988. — С. 89–115.
2. **Гимади Э. Х., Гончаров Е. Н., Залюбовский В. В., Пляскина Н. И., Харитонова В. Н.** О программно-математическом обеспечении для задачи ресурсно-календарного планирования Восточно-Сибирского нефтегазового комплекса // Вестн. НГУ. Сер. Математика, механика, информатика. — 2010. — Т. 10, вып. 4. — С. 51–67.
3. **Гимади Э. Х., Залюбовский В. В., Севастьянов С. В.** Полиномиальная разрешимость задач календарного планирования со складируемыми ресурсами и директивными сроками // Дискрет. анализ и исслед. операций. Сер. 2. — 2000. — Т. 7, №1. — С. 9–34.
4. **Гимади Э. Х., Пузынина Н. М., Севастьянов С. В.** О некоторых экстремальных задачах реализации крупных проектов типа БАМ // Экономика и мат. методы. — 1979. — Т. 15, вып. 5. – С. 1017–1021.
5. **Кочетов Ю. А., Столяр А. А.** Использование чередующихся окрестностей для приближённого решения задачи календарного планирования с ограниченными ресурсами // Дискрет. анализ и исслед. операций. Сер. 2. — 2003. — Т. 10, №2. — С. 29–55.

6. **Кочетов Ю. А., Столяр А. А.** Новые жадные эвристики для задачи календарного планирования с ограниченными ресурсами // Дискрет. анализ и исслед. операций. Сер. 2. — 2005. — Т. 12, № 1. — С. 12–36.
7. **Blaźewicz J., Lenstra J. K., Rinnoy Kan A. H. G.** Scheduling subject to resource constraints: classification and complexity // Discrete Appl. Math. — 1983. — Vol. 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 mode version // Eur. J. Oper. Res. — 2003. — Vol. 149, N 2. — P. 268–281.
9. **Brucker P., Drexl A., Möhring R., Neumann K., Pesch E.** Resource-constrained project scheduling: notation, classification, models, and methods // Eur. J. Oper. Res. — 1999. — Vol. 112, N 1. — P. 3–41.
10. **Coelho J., Tavares L.** Competitive analysis of metaheuristics for the resource constrained project scheduling problem // Tech. Report, Department of Civil Engineering, Instituto Superior Tecnico, Portugal, 2003.
11. **Debels D., De Reyck Leus B. R., Vanhoucke M.** A hybrid scatter search/electromagnetism meta-heuristic for project scheduling // Eur. J. Oper. Res. — 2006. — Vol. 169. — P. 638–653.
12. **Debels D., Vanhoucke M.** Decomposition-based genetic algorithm for the resource-consrtained project scheduling problem // Oper. Res. — 2007. — Vol. 55. — P. 457–469.
13. **Gimadi E. Kh., Sevastianov S. V.** On solvability of the project scheduling problem with accumulative resources of an arbitrary sign // Proc. Oper. Res., 2002. — Berlin; Heidelberg; New York: Springer-Verl., 2003. — P. 241–246.
14. **Goncharov E. N.** A greedy heuristic approach for the resource-constrained project scheduling problem // Stud. Informatica Universalis. — 2012. — Vol. 9, N 3. — P. 79–90.
15. **Hartmann S.** A competitive genetic algorithm for resource-constrained project scheduling // Naval Res. Logist. — 1998. — Vol. 45. — P. 733–750.
16. **Hartmann S.** A self-adapting genetic algorithm for project scheduling under resource constraints // Naval Res. Logist. — 2002. — V. 49. — P. 433–448.
17. **Hartmann S., Briskorn D.** A survey of variants and extentions of the resource-constrained project scheduling problem // Eur. J. Oper. Res. — 2010. — Vol. 207. — P. 1–14.
18. **Kolisch R.** Efficient priority rules for the resource constrained project scheduling problem // J. Oper. Manage. — 1996. — Vol. 14. — P. 179–192.
19. **Kolisch R.** Serial and parallel resource-constrained project scheduling methods revisited: theory and computation // Eur. J. Oper. Res. — 1996. — Vol. 90. — P. 320–333.
20. **Kolisch R., Drexl A.** Adaptive search for solving hard project scheduling problems // Naval Res. Logist. — 1996. — Vol. 43, N 1. — P. 23–40.
21. **Kolisch R., Hartmann S.** Experimental investigation of heuristics for resource-constrained project scheduling: an update // Eur. J. Oper. Res. —

2006. — Vol. 174. — P. 23–37.

22. **Kolisch R., Hartmann S.** Heuristic algorithms for solving the resource-constrained project scheduling problem: classification and computational analysis // Project scheduling: recent models, algorithms and applications. — Berlin: Kluwer Acad. Publ., 1999. — P. 147–178.
23. **Kolisch R., Sprecher A.** PSPLIB — a project scheduling problem library // Eur. J. Oper. Res. — 1996. — Vol. 96, N 1. — P. 205–216.
24. **Li R. Y., Willis J.** An iterative scheduling technique for resource-constrained project scheduling // Eur. J. Oper. Res. — 1992. — Vol. 56. — P. 370–379.
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. — P. 557–588.
26. **Ozdamar L., Ulusoy G.** A note on an iterative forward/backward scheduling technique with reference to a procedure by Li and Willis // Eur. J. Oper. Res. — 1996. — Vol. 89. — P. 400–407.
27. **Proon S., Jin M.** A genetic algorithm with neighborhood search for the resource-constrained project scheduling problem // Naval Res. Logist. — 2011. — Vol. 58. — P. 73–82.
28. **Schirmer A.** Case-based reasoning and improved adaptive search for project scheduling // Naval Res. Logist. — 2000. — Vol. 47, N 3. — P. 201–222.
29. **Tormos P., Lova A.** Integrating heuristics for resource-constrained project scheduling: One step forward // Tech. Report, Department of Statistics and Operations Research, Universidad Politecnica de Valencia, 2003.
30. **Tormos P., Lova A.** An efficient multi-pass heuristic for project scheduling with constrained resources // Int. J. Production Res. — 2003. — Vol. 41, N 5. — P. 1071–1086.
31. **Tormos P., Lova A.** A competitive heuristic solution techniques for resource-constrained project scheduling // Ann. Oper. Res. — 2001. — Vol. 102. — P. 65–81.
32. **Valls V., Ballestin F., Quintanilla S.** Justification and RCPSP: a technique that pays // Eur. J. Oper. Res. — 2005. — Vol. 165, N 2. — P. 375–386.
33. **Valls V., Ballestin F., Quintanilla S.** A hybrid genetic algorithm for the resource-constrained project scheduling problem // Eur. J. Oper. Res. — 2008. — Vol. 185, N 2. — P. 495–508.

Гончаров Евгений Николаевич,
e-mail: gon@math.nsc.ru

Статья поступила
30 августа 2013 г.

Переработанный вариант —
29 января 2014 г.