ДИСКРЕТНЫЙ АНАЛИЗ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ
Октябрь–декабрь 2022. Т. 29, № 4. С. 15–37

УДК 519.8+518.25 DOI 10.33048/daio.2022.29.734

АЛГОРИТМ ЛОКАЛЬНОГО ПОИСКА ДЛЯ ЗАДАЧИ КАЛЕНДАРНОГО ПЛАНИРОВАНИЯ С ОГРАНИЧЕННЫМИ РЕСУРСАМИ

Е. Н. Гончаров[1,2]

[1] Институт математики им. С. Л. Соболева,
пр. Акад. Коптюга, 4, 630090 Новосибирск, Россия
[2] Новосибирский гос. университет,
ул. Пирогова, 2, 630090 Новосибирск, Россия
E-mail: gon@math.nsc.ru

Аннотация. Рассматривается задача календарного планирования с ограниченными ресурсами по критерию минимизации длины расписания. Все ресурсы возобновимы, прерывания работ запрещены. Предложен алгоритм локального поиска, использующий список запретов и два типа окрестностей. Численный эксперимент на примерах из библиотеки PCPLIB показал конкурентоспособность предложенного алгоритма. Были получены одни из лучших средних отклонений найденных решений от величины критического пути, для нескольких примеров из серии тестовых примеров j120 найдены лучшие (ранее неизвестные) решения. Табл. 4, библиогр. 47.

Ключевые слова: задача календарного планирования с ограниченными ресурсами, возобновимые ресурсы, локальный поиск с запретами, VNS, PSPLIB.

Введение

Рассматривается задача календарного планирования в условиях ограниченных ресурсов (RCPSP, resource-constrained project scheduling problem). Задача состоит в минимизации общего времени выполнения проекта с учётом отношения предшествования работ и выполнения ресурсных ограничений. По классификации, предложенной в [1], эта задача обозначается через $\mathrm{PS}|\mathrm{prec}|C_{\max}$.

Исследование выполнено в рамках государственного задания ИМ СО РАН (проект № FWNF–2022–0019).

Дано множество работ, частичный порядок на этом множестве задаётся ациклическим ориентированным графом. Для каждой работы известны длительность её выполнения, множество потребляемых ею ресурсов и их объёмы. Профиль потребления ресурсов на протяжении выполнения работы полагается постоянным. Считаем, что в каждый единичный интервал горизонта планирования $\widehat{T}$ для каждого ресурса выделяется одинаковое его количество. За пределами горизонта планирования ресурсы не ограничены.

Все ресурсы возобновимы. Прерывания работ не допускаются. Необходимо найти расписание выполнения работ с минимальным сроком завершения проекта. При этом должны быть учтены технологические ограничения предшествования работ и ограничения на ресурсы.

Эта задача принадлежит к классу NP-трудных задач [2]. Заметим, что замена свойства возобновимости свойством складируемости ограниченных ресурсов приводит к полиномиальной разрешимости рассматриваемой задачи (см. [3, 4]).

Точными методами, такими как математическое программирование или алгоритмы переборного типа, удаётся решать лишь примеры сравнительно небольшой размерности. Данное обстоятельство послужило мотивацией к разработке методов решения задачи RCPSP, основанных на эвристиках и метаэвристиках, позволивших решать задачи более значительной размерности.

На протяжении нескольких последних десятилетий применение задачи RCPSP привлекает всё больший интерес со стороны исследователей, о чём свидетельствуют многочисленные обзоры по методам её решения. Отметим лишь некоторые из них: [5–9].

Среди различных эвристических методов решения RCPSP существенное место занимают алгоритмы локального поиска. Эти методы обеспечивают в большинстве случаев решения лучше, чем методы построения, поскольку они выполняют процедуру поиска, стартуя от некоторого допустимого расписания, сгенерированного одним или несколькими методами построения. В этой статье представлен подход, сочетающий в себе концепции Tabu search и алгоритма поиска по переменной окрестности, который использует кодирование последовательности работ в качестве основы для локального поиска по окрестности. Эта стратегия позволяет быстро вычислять хорошие расписания в режиме реального времени. Алгоритм поиска в окрестности (NS) используется для улучшения одного допустимого решения путём фиксирования времени начала выполнения некоторых работ и изменения расписания других работ. В [10] предложено пять таких методов. Численные эксперименты показали, что метод отбора «Блок» явно превосходит другие методы. Этот метод использует следующий способ построения соседнего расписания. В исходном

допустимом расписании некоторым способом выделяются сразу целая группа работ — блок, затем в соответствии с некоторым алгоритмом производится изменение расписания сразу для всех работ этого блока. После этого проводится достройка решения, которое и принимается за соседнее. В представленном в данной работе алгоритме локального поиска применяется только метод выбора блоков для формирования подзадачи изменения расписания для работ из этого блока во всех NS-операторах. При этом используется два альтернативных варианта окрестностей. Одна из окрестностей является модификацией окрестности, предложенной в [10], а другая — модификацией окрестности, предложенной в [11]. Качество алгоритма было протестировано на примерах из известной библиотеки PSPLIB [18]. Для тестирования были использованы множества примеров j60, j90 и j120 размерностью 60, 90 и 120 работ соответственно. Результаты численных экспериментов приводятся в данной работе. Показано, что в результате проведённых численных экспериментов получены одни из лучших значений средних отклонений решений от величины критического пути, и приведено их сравнение с другими результатами. Выявлены также группы примеров, на которых алгоритм имеет преимущество в сравнении с другими алгоритмами. Для восьми примеров из j120 были найдены лучшие (ранее неизвестные) эвристические решения.

Предварительная версия статьи опубликована в материалах международной конференции «Теория математической оптимизации и исследование операций» 2019 г. [12]. Данная версия статьи включает полное описание алгоритма, значительно расширен численный эксперимент, дополнен список литературы.

1. Постановка задачи

Задача RCPSP может быть сформулирована следующим образом. Будем рассматривать проект как ориентированный ациклический граф $G = (N, A)$, где N является множеством работ проекта, а A представляет собой множество пар работ, между которыми установлены отношения предшествования. Обозначим через $N = \{1, \ldots, n\} \cup \{0, n+1\}$ множество работ, в котором работы 0 и $n+1$ фиктивны, они задают начало и завершение всего комплекса работ. Все другие работы будем называть фактическими. Отношение предшествования на множестве N зададим множеством пар $A = \{(i, j) \mid i - \text{предшественник } j\}$. Если $(i, j) \in A$, то работа j не может начаться раньше завершения работы i. В множестве A содержатся все пары $(0, j)$ и $(j, n+1)$, $j = 1, \ldots, n$.

Через K обозначим множество типов ресурсов, необходимых для выполнения работ. Горизонт планирования $\widehat{T}$ делится на временны́е интервалы равной длины $[t-1, t)$, $t = 1, \ldots, \widehat{T}$. Все ресурсы предполагаются

возобновимыми, т. е. в каждый единичный временно́й интервал горизонта планирования $\widehat{T}$ для каждого типа ресурсов выделяется фиксированный объём этого ресурса $R_k \in Z^+$.

Фактическая работа j имеет детерминированную длительность своего выполнения $d_j \in Z^+$ и требует в каждую единицу времени своего выполнения $r_{jk} \geqslant 0$ единиц ресурса типа $k \in K$. В целях избежания неразрешимости задачи будем считать, что $r_{jk} \leqslant R_k$, $j \in N$, $k \in K$. Фиктивные работы 0 и $n+1$ имеют нулевую длительность и нулевое потребление ресурсов. Фактические работы имеют ненулевую длительность.

Введём переменные задачи. Через $s_j \geqslant 0$ обозначим момент начала выполнения j-й работы. Так как работы выполняются без прерывания, момент окончания j-й работы определяется равенством $c_j = s_j + d_j$. Определим расписание S как $(n+2)$-вектор $(s_0, \ldots, s_{n+1})$. Время завершения проекта $T(S)$ соответствует моменту завершения последней работы проекта, т. е. $T(S) = c_{n+1}$. Через $J(t) = \{j \in N \mid s_j \leqslant t < c_j\}$ обозначим множество работ, выполняемых в единичном интервале времени $[t-1, t)$ при расписании S. Задача заключается в нахождении допустимого расписания $S = \{s_j\}$ с минимальным временем завершения проекта $T(S)$. Её можно формализовать следующим образом: минимизировать время завершения проекта

$$T(S) = \max_{j \in N}(s_j + p_j) \tag{1}$$

при ограничениях

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

$$\sum_{j \in J(t)} r_{jk} \leqslant R_k, \quad k \in K,\ t = 1, \ldots, \widehat{T}, \tag{3}$$

$$s_j \in \mathbf{Z}^+, \quad j \in N. \tag{4}$$

Неравенства (2) определяют ограничения предшествования работ. Соотношение (3) обеспечивает соблюдение ограничений по возобновимым ресурсам: суммарное количество ресурса типа k, потребляемое всеми работами, выполняемыми в единичном интервале времени $[t-1, t)$, не должно превышать имеющегося в наличии количества этого ресурса в данный момент времени. Наконец, (4) определяет искомые переменные.

2. Алгоритм локального поиска

2.1. Представление решений. Каждое допустимое решение задачи будем представлять в виде списка работ $L = (j_0, \ldots, j_{n+1})$ [13].

Предполагается, что все рассматриваемые списки совместимы с отношениями предшествования, т. е. каждая работа в таком списке не может находиться ранее любого из своих предшественников. Расписание

называется активным, если время начала выполнения работ таково, что никакие работы не могут быть начаты раньше времени начала своего выполнения, не нарушая при этом ни отношения предшествования работ, ни ресурсные ограничения.

Известны алгоритмы последовательного и параллельного декодирования, позволяющие по допустимому списку работ строить активное расписание. Для произвольного списка L процедура последовательного декодирования (S-SGS) вычисляет активное расписание $S(L)$ [13]. Известно, что среди активных расписаний есть оптимальное. Параллельный декодер (P-SGS) последовательно рассматривает возрастающие моменты времени и выявляет подмножество подходящих работ, чтобы начать их выполнение в тот или иной момент.

2.2. Эвристическое правило определения степени дефицитности ресурсов. Для определения степени дефицитности ресурсов будем использовать следующее эвристическое правило. На предварительном этапе работы алгоритма, ещё до начала процедуры локального поиска, определим сравнительную степень дефицитности ресурсов. Для этого ранжируем их по уменьшению дефицитности на основе решения релаксированной задачи, и будем предполагать, что и в исходной задаче RCPSP этот порядок будет сохраняться. Назначим каждому из типов ресурсов вес $w_k, k = 1, \ldots, K$, в соответствии с этим ранжированием. Имея такие веса, можно будет в ходе процедуры локального поиска отдавать предпочтение тем перестановкам в окрестности, которые наиболее оптимальным образом используют более дефицитные ресурсы. Обозначим через v_j вес работы j, $v_j = \sum\limits_{k \in K} r_{jk} w_k / R_k$.

Рассмотрим релаксированную задачу, в которой условие возобновимости ресурсов ослаблено до условия их складируемости:

$$T(S) = \max_{j \in N} (s_j + p_j) \tag{5}$$

при ограничениях

$$s_i + p_i \leqslant s_j, \quad (i, j) \in A, \tag{6}$$

$$\sum_{t'=1}^{t} \sum_{j \in J(t')} r_{jk} \leqslant \sum_{t'=1}^{t} R_k, \quad k \in K,\ t = 1, \ldots, \widehat{T}, \tag{7}$$

$$s_j \in \mathbf{Z}^+, \quad j \in N. \tag{8}$$

Для задачи (5)–(8) известен быстрый полиномиальный приближённый алгоритм [3]. Этот алгоритм асимптотически точный с абсолютной погрешностью, стремящейся к нулю с ростом размерности задачи, его вычислительная сложность зависит от числа работ n как функция

порядка n^2. Кроме того, известен также точный полиномиальный алгоритм в случае, когда длительности работ целочисленны [4, 14].

Так как в исходной задаче (1)–(4) длительности работ целочисленны, возможно использование любого из этих двух алгоритмов. В данной статье использовался первый алгоритм. В результате работы этого алгоритма помимо приближённого решения релаксированной задачи (5)–(8) получаем также и неиспользованные остатки всех складируемых ресурсов в момент завершения работы алгоритма. Эти остатки ресурсов и позволяют ранжировать их в порядке степени дефицитности: чем меньше этот остаток, тем ресурс дефицитнее. Найденное таким образом ранжирование ресурсов для задачи (5)–(8) будем применять и для задачи (1)–(4), делая предположение, что такой порядок дефицитности ресурсов сохранится и для возобновимых ресурсов.

2.3. Блок работ. Для заданного допустимого расписания $S = (s_0, \ldots, s_{n+1})$ и корневой работы $j = 1, \ldots, n$ оператор NS будет пытаться отыскать новое расписание для работ из блока работ A_j^s. При этом для всех других работ, не входящих в блок A_j^s, их расписание не изменится. Пусть P — заранее определённое число работ, которые будут включены в блок A_j^s. Значение P влияет на вычислительное время для получения решения окрестности путём перепланирования блока работ. Меньшее значение P означает меньшее число работ, которые необходимо перепланировать, и меньше времени для получения нового расписания. Будем использовать следующий метод выбора блока для создания A_j^s [10].

CreateBlock$(j, S) \to A_j^s$

Шаг 1. $A_j^s = \{j\}$, $b = 0$, создаём случайный список для всех работ из множества $A/\{j\}$. Пусть работа i — первая в этом списке.

Шаг 2. Если $s_j - p_i - b \leqslant s_i \leqslant s_j + p_j + b$, то $A_j^s := A_j^s \cup \{i\}$.

Шаг 3. Если $|A_j^s| = P$, то идём на шаг 6.

Шаг 4. Если i является последней работой из тех, что не принадлежат множеству A_j^s в соответствии с порядком, определённым на шаге 1, то $b := b + 1$.

Шаг 5. Пусть i будет следующей работой среди тех, которые не принадлежат A_j^s в соответствии с порядком, определённым на шаге 1. Переходим на шаг 2.

Шаг 6. Конец.

Приведённый алгоритм выбирает множество из P работ, которые перекрываются с работой j или близки к ней при заданном допустимом расписании.

2.4. Первоначальное решение. Выбор стартового решения и алгоритма для его нахождения не критичен для локальных методов поиска. В [15] отмечено, что существует некоторая дилемма, касающаяся выбора исходного решения, используемого алгоритмами NS. С одной стороны, если алгоритм NS стартует с очень хорошего решения, то сужается пространство для поиска его значительного улучшения. С другой стороны, для улучшения плохого начального решения может потребоваться длительное вычислительное время. Мы можем использовать любой доступный метод для быстрого создания хорошего начального решения, например, схемы S-SGS и P-SGS, стохастические методы с процедурой forward-backward improvement procedure (FBI), жадные алгоритмы. В этой статье использован стохастический жадный алгоритм для создания стартового решения.

2.5. Список запретов. Метод NS использует знания, полученные в результате рассмотренных ранее решений. Эти знания хранятся в списке запретов, используемом в качестве памяти для того, чтобы избежать цикличности, т. е. повторения недавних преобразований, применённых для получения оцениваемых решений. Недавнее пребывание решения в табу-списке регистрируется путём сохранения в каждой записи табу-списка атрибутов последних посещённых решений. Мы используем табу-список постоянной длины. В качестве запретного статуса произвольного решения L рассматриваем сумму начальных времён

$$TS(L) = \sum_{j=1}^{n} s_j$$

для расписания $S(L)$.

2.6. Окрестность A. Первая окрестность $N_A(S)$ является модификацией схемы, предложенной в [16]. Для заданного допустимого расписания $S = \{s_0, \ldots, s_{n+1}\}$, корневой работы $j \in A$ и параметра P определяем блок работ A_j^s. Оператор NS производит перепланировку множества работ A_j^s , в то время как моменты начала всех других работ остаются неизменными. Подзадача перепланирования работ из A_j^s представляется следующим образом.

Фиксируем время начала всех действий, не относящихся к множеству A_j^s, и освобождаем ресурсы, используемые всеми работами из A_j^s в каждый период времени t. Доступный объём ресурсов типа k для работ из A_j^s в момент t есть R_k минус ресурс, используемый всеми работами, не относящимися к множеству A_j^s в момент времени t. Затем находим самое раннее время начала (EST) и самое позднее время окончания (LFT) для каждой работы из $i \in A_j^s$ следующим образом:

$$\mathrm{EST}_i = \max\{s_l + p_l \mid l \notin A_j^s,\ (l,i) \in A\},$$
$$\mathrm{LFT}_i = \max\{s_l \mid l \notin A_j^s,\ (l,i) \in A\}.$$

Интервал $(\mathrm{EST}_i, \mathrm{LFT}_i)$ задаёт временно́е окно для работы i, в пределах которого эта работа может быть перепланирована так, чтобы новое расписание оставалось допустимым для всех работ, которые не были перепланированы. Задача перепланирования состоит в том, чтобы перепланировать все работы из A_j^s так, чтобы минимизировать их общее время выполнения, сохраняя при этом ресурсные ограничения каждого момента времени и ограничения временно́го окна.

Для получения оптимального решения проблемы перепланирования в [10] использовался коммерческий пакет программ для задачи целочисленного линейного программирования. В [16] для решения этой задачи применялась последовательная схема генерации расписаний (S-SGS, serial schedule generation scheme) [13]. В данной статье предлагаем модификацию этого алгоритма. На каждой итерации создаём новый случайный вектор для работ $i \in A_j^s$ в качестве списка приоритетов. Затем упорядочиваем работы в A_j^s по невозрастанию их весов v_j. Вектор создаётся итеративно путём случайного выбора следующей работы из упорядоченного списка среди всех невыбранных работ, чьи предшественники в A_j^s были выбраны. Согласно списку приоритетов работы одна за одной перемещаются на самое раннее (последнее) время их начала, которое допустимо по ограничениям предшествования и ресурсным ограничениям, и помещаются во временно́е окно $(\mathrm{EST}_i, \mathrm{LFT}_i)$. Как только все работы $i \in A_j^s$ будут перепланированы, добавляем работы, которые не принадлежат A_j^s, чтобы сформировать полное допустимое решение. Затем выполняется глобальный сдвиг влево для всех работ в A, чтобы, возможно, сократить общее время выполнения всех работ. Полученное новое расписание сравнивается с предыдущим решением (перед применением оператора NS). Если время выполнения проекта улучшено, то полученное расписание заменяет предыдущее рекордное (лучшее среди найденных) расписание. Если улучшения не было достигнуто, то процедура S-SGS применяется к расписанию с новым списком случайных приоритетов в качестве следующей итерации. Это происходит до тех пор, пока количество итераций не достигнет заданного предела итераций λ.

2.7. Окрестность B. В качестве окрестности $N_B(S)$ будем использовать модификацию окрестности, предложенную в [11]. Для заданного списка работ L (и соответствующего активного расписания $S = \{s_0, \ldots, s_{n+1}\}$) и корневой работы $j \in A$ находим блок работ A_j^s. Если блок содержит хотя бы одного предшественника работы j, то полагаем A_j^s пустым. Представляем список L в виде трёх последовательных списков

$L = A^1, A_j^s, A^2$. Список L' получается из списка L с помощью следующих шагов. Фиксируем стартовое время всех работ из множества A^1 и учитываем ресурсы, используемые всеми работами из A^1 в каждом временно́м отрезке t. Находим частичное расписание для работ из A^1, используя последовательную схему генерации расписаний. Затем расширяем частичное расписание, добавляя в него работы, относящиеся к множеству A_j^s, с помощью процедуры параллельного декодирования. В соответствии с этой процедурой для каждого момента времени t есть соответствующее допустимое множество E_t, т. е. множество работ, которые могут быть запущены в момент t без нарушения каких-либо ограничений. Существует экспоненциально много возможностей выбрать подмножество работ из множества E_t для включения в расписание. С этой целью будем решать многомерную задачу о рюкзаке с целевой функцией, максимизирующей взвешенный коэффициент использования ресурсов [17]:

$$\max \sum_{j \in E_t} x_j \sum_{k \in K} \frac{w_k r_{jk}}{R_k}, \tag{9}$$

$$\sum_{j \in E_t} r_{jk} x_j \leqslant R_k - \sum_{j \in J(t)} r_{jk}, \quad k \in K, \tag{10}$$

$$x_j \in \{0, 1\}, \quad j \in N. \tag{11}$$

Правая часть ограничения (10) является остатком ресурса типа k в момент времени t. Для решения этой проблемы используем жадные рандомизированные процедуры адаптивного поиска (GRASP). Заметим, что в процессе решения проблемы (9)–(11) преимущество может быть получено не для тех работ, которые наилучшим образом используют ресурсы, а для тех, которые наилучшим образом используют более дефицитные ресурсы.

Новый список L' строится следующим образом. Помещаем работы $j \in A_j^s$ в список L' в порядке неубывания времени начала их выполнения в частичном расписании. Оставшиеся работы помещаются в список L' в том же самом порядке, как они были в списке L. Расписание $S(L')$ будем называть соседним для расписания S. Множество, содержащее все соседние расписания, называется окрестностью расписания S и обозначается через $N_B(S)$.

3. Схема алгоритма

Шаг 1. Создаём начальное расписание S и полагаем $T^* := T(S)$, $S^* := S$. Начальное расписание можно построить любым доступным образом, мы воспользуемся алгоритмом GRASP.

Шаг 2. Положим $P = 1$.

Шаг 3. До тех пор пока не будет удовлетворён критерий остановки, выполняем следующие шаги.

Шаг 3.1. Равновероятно выбираем тип окрестности (A или B).

Шаг 3.2. Строим соседнее решение S', не запрещённое Tabu-списком TL.

Шаг 3.3. Если $T(S') < T^*$, то полагаем $T^* := T(S')$, $S^* := S'$.

Шаг 3.4. Обновляем список запретов TL и полагаем $S := S'$.

Шаг 4. Если значение T^* не изменяется в течении определённого заранее количества итераций, то меняем мощность множества A_j^s. В зависимости от статистики предыдущей работы алгоритма NS эта мощность может быть либо увеличена, либо уменьшена. Если в течение заданного числа предыдущих итераций процент непустых соседних решений достаточно высок (предопределённый параметр), то увеличиваем параметр P на единицу. Если процент пустых соседей высок, то это означает, что мощность A_j^s слишком высока и в ней часто оказываются предшественники работ из этого множества. В этом случае уменьшаем параметр P на единицу.

Смысл шага 4 состоит в увеличении мощности множества A_j^s в надежде, что будут возникать новые решения. Однако такое увеличение можно проводить лишь до некоторого предела: когда в это множество будут почти всегда попадать работы-предшественники и оно вынуждено будет становиться пустым, необходимо уменьшать параметр P, тем самым вновь попадая в область непустых множеств A_j^s.

Шаг 5. Если изменение параметра P было произведено заранее предопределённое число раз, а значение T^* не изменилось, то полагаем $P = 1$ и заново назначаем весовые коэффициенты ресурсов w_k, $k = 1, \ldots, K$. Назначение производится случайным образом, в соответствии с произведённым ранжированием ресурсов. Счётчик числа корректировок параметра P зануляется.

Шаг 6. Наконец, если значение T^* не изменяется заданное число итераций, то создаём новое начальное расписание и возвращаемся на шаг 1.

В качестве критерия останова алгоритма будем рассматривать достижение максимального числа просмотренных решений λ. Значение T^* является результатом работы алгоритма.

4. Численные эксперименты

Алгоритм NS был закодирован на C++ в системе Visual Studio и реализован на компьютере с процессором 3,4 ГГц и 8 ГБ оперативной памяти под управлением операционной системы Windows 7. Для оценки

Таблица 1

Средние отклонения решений от критического пути для примеров из j60

Алгоритм	APD, %	
	$\lambda = 50000$	$\lambda = 500000$
Specialist GA [21]	**10,52**	**10,42**
GANS [16]	**10,52**	—
TS, VNS [эта статья]	10,55	10,44
Sequential(SS(FBI)) [22]	10,58	10,45
GH + SS(LS) [23]	10,54	10,46
AI(FBI) [24]	10,55	—
TS + SS(FBI) [25]	10,57	—
GA(FBI) [26]	10,57	—
GA(FBI) [27]	10,57	10,49
EA(GA(LS) + DEA(LS)) [28]	10,58	—
PSO(LS) [29]	10,62	—
GA [30]	10,63	10,51
Parallel(MA(LS)) [31]	10,63	—
PL(LS) [32]	10,64	—
GA(FBI) [33]	10,65	—
SFL(LS) [34]	10,66	—
GA(FBI) [35]	10,66	—
ACOSS [36]	10,67	—
GAPS [37]	10,67	10,67
GA [38]	10,68	—
Specialist(PSO(LS)) [39]	10,68	—
GA(LS) [41]	10,70	—
Scatter search – FBI [42]	10,71	10,53

производительности предложенного алгоритма были использованы примеры из электронной библиотеки тестовых примеров PSPLIB, представленной в [18]. Мы использовали наборы тестовых примеров j60, j90 и j120. Множества j60 и j90 содержат примеры с 60 и 90 работами в каждом соответственно, по 48 серий примеров в каждой, по 10 экземпляров в каждой серии, всего по 480 примеров в каждом из множеств. Набор данных j120 содержит 60 серий тестовых примеров, по 10 экземпляров в каждой серии, всего 600 экземпляров. Эти примеры доступны по ссылке `www.om-db.wi.tum.de/psplib/data.html` вместе с наилучшими найденными решениями, с указанием авторов этих решений и времени их представления в библиотеку.

В каждом примере рассматривается четыре типа ресурсов. Содержание примера определяется тремя параметрами: сложность сети (NC),

Таблица 2

Средние отклонения решений от критического пути для примеров из j90

Алгоритм	APD, %	
	$\lambda = 50000$	$\lambda = 500000$
GA [38]	9,90	—
Sequential(SS(FBI)) [22]	9,96	9,74
TS, VNS [эта статья]	9,98	9,78
Sequential(SS) [43]	10,04	—
SS(EM + FBI) [42]	10,09	9,80
PL(LS) [44]	11,60	—
TS [45]	12,15	—

коэффициент ресурсов (RF) и ресурсный потенциал (RS). NC определяет среднее число предшественников для каждой работы. Параметр RF устанавливает средний процент количества типов ресурсов, необходимых для выполнения каждой работы. Параметр RS устанавливает средний процент объёма выделяемых ресурсов в каждый момент времени. Нулевое значение коэффициента RS соответствует минимальной потребности в ресурсах каждого типа для выполнения всех работ, в то время как RS = 1 соответствует случаю неограниченных ресурсов. Значения параметров, использованные для создания примеров из набора j120, следующие: NC $\in \{1{,}5; 1{,}8; 2{,}1\}$, RF $\in \{0{,}25; 0{,}5; 0{,}75; 1\}$ и RS $\in \{0{,}1; 0{,}2; 0{,}3; 0{,}4; 0{,}5\}$. Известно [19], что значения параметров RF = 4, RS = 0,2 соответствуют достаточно сложным сериям. Примеры j12016, j12036, j12056, j12011, j12031, j12051 с $n = 120$ являются наиболее трудными, т. е. имеют наибольший размер между найденными наилучшими решениями и длиной критического пути. Каждый триплет таких идентификаторов соответствует значениям NC $\in \{1{,}5; 1{,}8; 2{,}1\}$ соответственно.

Более подробное описание процесса создания примеров можно найти в [20].

Эффективность эвристических алгоритмов принято оценивать, сравнивая полученные ими решения и ограничивая при этом суммарное число проведённых алгоритмом итераций. Для проведения численных экспериментов использованы $\lambda = 50000$ и $\lambda = 500000$. Показателем качества решений является их среднее процентное отклонение (APD, average percent deviation) от нижних границ, в качестве которых используется величина критического пути [20].

В табл. 1–3 приведено сравнение средних отклонений полученных длин расписаний от величин критического пути (в процентах) для примеров серий j60, j90 и j120 соответственно из библиотеки PSPLIB. Из этих

Таблица 3

Средние отклонения решений от критического пути для примеров из j120

Алгоритм	APD, %	
	$\lambda = 50000$	$\lambda = 500000$
Specialist GA [21]	30,50	**29,74**
TS,VNS [эта статья]	30,56	29,88
GA [30]	30,66	29,91
biased random-key GA [27]	32,76	30,08
GANS [16]	**30,45**	30,78
ACOSS [36]	30,56	—
DBGA [38]	30,69	—
GH + SS(LS) [23]	30,78	30,39
GA [38]	30,82	—
PL(LS) [32]	31,02	—
SFL(LS) [34]	31,11	—
Sequential(SS(FBI)) [22]	31,16	30,39
EA(GA(LS) + DEA(LS)) [28]	31,22	—
Specialist(PSO(LS)) [39]	31,23	—
GA – Hybrid, FBI [40]	31,24	30,95
GA(FBI) [26]	31,28	—
GA(FBI) [33]	31,30	—
Enhanced SS [25]	31,37	—
GA(LS) [46]	31,38	—
GA(LS) [41]	31,40	—
Scatter search – FBI [42]	31,57	30,48
GAPS [37]	31,44	31,20
GA, FBI [47]	31,58	—

таблиц видим, что предложенный алгоритм позволяет получать сравнительно хорошие решения и для всех трёх серий примеров он занимает лидирующие позиции.

Помимо этого для 8 примеров из множества примеров j120 удалось отыскать наилучшие (ранее неизвестные) эвристические решения. Все эти решения представлены в библиотеке PSPLIB. В табл. 4 дан список этих примеров, а также приведено процентное отклонение полученных решений от величины критического пути. Как видим, все полученные рекордные решения имеют такое отклонение от 100% и выше. Это свидетельствует о том, что алгоритм особенно хорошо себя показал на классе трудных примеров, имеющих большое отклонение решений от величины критического пути. Все рекордные решения были получены на примерах из серий 11, 16, 31, 36, 51 и 56, все эти серии были ранее отмечены как трудные серии.

Таблица 4

Список примеров из набора j120, для которых получены новые эвристические решения

Серия	Пример	APD примера, %
11	3	117,2
16	8	151,9
31	4	94,6
36	3	150,5
51	3	135,5
51	4	131,9
56	7	141,0
56	8	190,9

Заключение

В работе предложен алгоритм локального поиска с запретами для задачи календарного планирования с ограниченными ресурсами в соответствии с критерием минимизации времени выполнения всего проекта. При организации локального поиска он использует список запретов, а также два типа окрестностей. Алгоритм применяет эвристику, учитывающую степень критичности (дефицитности) ресурсов, которая является производной от решения ослабленной задачи с ограничением на совокупные ресурсы. Проведены численные эксперименты на примерах из электронной библиотеки PSPLIB. Их результаты показывают, что предлагаемый алгоритм является конкурентоспособной эвристикой. Были получены лучшие средние отклонения получаемых решений от нижней оценки, в качестве которой принимается величина критического пути. Для некоторых примеров из множества примеров j120 были улучшены лучшие из известных эвристических решений.

Дальнейшие исследования будут сосредоточены на построении гибридных алгоритмов для рассматриваемой задачи.

ЛИТЕРАТУРА

1. **Brucker P., Drexl A., Möhring R., Neumann K., Pesch E.** Resource-constrained project scheduling: Notation, classification, models, and methods // Eur. J. Oper. Res. 1999. V. 112, No. 1. P. 3–41.
2. **Blazewicz J., Lenstra J. K., Rinnoy Kan A. H. G.** Scheduling subject to resource constraints: Classification and complexity // Discrete Appl. Math. 1983. V. 5, No. 1. P. 11–24.

3. **Гимади Э. Х.** О некоторых математических моделях и методах планирования крупномасштабных проектов // Модели и методы оптимизации. Тр. Ин-та математики. Т. 10. Новосибирск: Наука, 1988. С. 89–115.
4. **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.
5. **Pellerin R., Perrier N., Berthaut F.** LSSPER: A survey of hybrid metaheuristics for the resource-constrained project scheduling problem // Eur. J. Oper. Res. 2020. V. 280, No. 2. P. 395–416.
6. **Abdolshah M.** A review of resource-constrained project scheduling problems (RCPSP) approaches and solutions // Int. Trans. J. Eng. Manage. Appl. Sci. Technol. 2014. V. 5, No. 4. P. 253–286.
7. **Vanhoucke M.** Resource-constrained project scheduling // Project Management with Dynamic Scheduling. Heidelberg: Springer, 2012. P. 107–137.
8. **Hartmann S., Briskorn D.** A survey of variants and extensions of the resource-constrained project scheduling problem // Eur. J. Oper. Res. 2010. V. 207, No. 1. P. 1–14.
9. **Kolisch R., Hartmann S.** Experimental investigation of heuristics for resource-constrained project scheduling: An update // Eur. J. Oper. Res. 2006. V. 174, No. 1. P. 23–37.
10. **Palpant M., Artigues C., Michelon P.** Solving the resource-constrained project scheduling problem with large neighborhood search // Ann. Oper. Res. 2004. V. 131. P. 237–257.
11. **Kochetov Yu. A., Stolyar A. A.** Evolutionary local search with variable neighborhood for the resource constrained project scheduling problem // Proc. 5th Int. Workshop Comput. Sci. Inf. Technol. (Ufa, Russia, Sept. 16–18). Ufa: USATU, 2003. P. 96–99.
12. **Goncharov E. N.** Variable neighborhood search for the resource constrained project scheduling problem // Mathematical Optimization Theory and Operations Research. Rev. Sel. Pap. 18th Int. Conf. (Yekaterinburg, Russia, July 8–12, 2019). Cham: Springer, 2021. P. 39–50. (Commun. Comput. Inf. Sci.; V. 1090).
13. **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. Boston: Kluwer Acad. Publ., 1999. P. 147–178.
14. **Gimadi E. Kh., Goncharov E. N., Mishin D. V.** On some implementations of solving the resource-constrained project scheduling problem // Yugoslav. J. Oper. Res. 2019. V. 29, No. 1. P. 31–42.
15. **Gagnon M., Boctor F. F., d'Avignon G.** A tabu search algorithm for the resource-constrained project scheduling problem // Proc. 2004 Conf. Adm. Sci. Assoc. Canada. (Quebec, Canada, June 5–8, 2004). Waterloo, ON: ASAC, 2004.

16. **Proon S., Jin M.** A genetic algorithm with neighborhood search for the resource-constrained project scheduling problem // Naval Res. Logist. 2011. V. 58. P. 73–82.
17. **Valls V., Ballestin F., Quintanilla S.** A population-based approach to the resource-constrained project scheduling problem // Ann. Oper. Res. 2004. V. 131. P. 305–324.
18. **Kolisch R., Sprecher A.** PSPLIB — a Project Scheduling Problem Library // Eur. J. Oper. Res. 1996. V. 96, No. 1. P. 205–216.
19. Project scheduling: Recent models, algorithms and applications. Boston: Kluwer Acad. Publ., 1999.
20. **Kolisch R., Sprecher A., Drexl A.** Characterization and generation of a general class of resource-constrained project scheduling problems // Manage. Sci. 1995. V. 41, No. 10. P. 1693–1703.
21. **Гончаров Е. Н., Леонов В. В.** Генетический алгоритм для задачи календарного планирования с ограниченными ресурсами // Автоматика и телемеханика. 2017. № 6. С. 173–189.
22. **Berthaut F., Pellerin R., Hajji A., Perrier N.** A path relinking-based scatter search for the resource-constrained project scheduling problem // Int. J. Proj. Organ. Manage. 2018. V. 10, No. 1. P. 1–36.
23. **Paraskevopoulos D. C., Tarantilis C. D., Ioannou G.** Solving project scheduling problems with resource constraints via an event list-based evolutionary algorithm // Expert Syst. Appl. 2012. V. 39, No. 4. P. 3983–3994.
24. **Mobini M., Mobini Z., Rabbani M.** An artificial immune algorithm for the project scheduling problem under resource constraints // Appl. Soft Comput. 2011. V. 11, No. 2. P. 1975–1982.
25. **Mahdi Mobini M. D., Rabbani M., Amalnik M. S., Razmi J., Rahimi-Vahed A. R.** Using an enhanced scatter search algorithm for a resource-constrained project scheduling problem // Soft Comput. 2009. V. 13. P. 597–610.
26. **Wang H., Li T., Lin T.** Efficient genetic algorithm for resource-constrained project scheduling problem // Trans. Tianjin Univ. 2010. V. 16, No. 5. P. 376–382.
27. **Goncalves J., Resende M. G. C, Mendes J.** A biased random key genetic algorithm with forward-backward improvement for resource-constrained project scheduling problem // J. Heuristics. 2011. V. 17. P. 467–486.
28. **Elsayed S., Sarker R., Ray T., Coello C. C.** Consolidated optimization algorithm for resource-constrained project scheduling problems // Inf. Sci. 2017. V. 418–419. P. 346–362.
29. **Czogalla J., Fink A.** Particle swarm topologies for resource constrained project scheduling // Nature Inspired Cooperative Strategies for Optimization (NICSO 2008). Heidelberg: Springer, 2009. P. 61–73.
30. **Lim A., Ma H., Rodrigues B., Tan S. T., Xiao F.** New meta-heuristics for the resource-constrained project scheduling problem // Flex. Services Manuf. J. 2013. V. 25, No. 1–2. P. 48–73.

31. Chen D., Liu S., Qin S. Memetic algorithm for the resource-constrained project scheduling problem // Proc. 11th World Congr. Intell. Control Autom. (WCICA 2014) (Shenyang, China, June 27–30, 2014). Piscataway: IEEE, 2014. P. 4991–4996.

32. Zheng X., Wang L. A multi-agent optimization algorithm for resource constrained project scheduling problem // Expert Syst. Appl. 2015. V. 42, No. 15–16. P. 6039–6049.

33. Zamani R. A competitive magnet-based genetic algorithm for solving the resource-constrained project scheduling problem // Eur. J. Oper. Res. 2013. V. 229, No. 2. P. 552–559.

34. Zheng X., Wang L. An effective shuffled frog-leaping algorithm for resource-constrained project scheduling problem // Comput. Oper. Res. 2012. V. 39, No. 5. P. 890–901.

35. Ismail I. Y., Barghash M. A. Diversity guided genetic algorithm to solve the resource constrained project scheduling problem // Int. J. Plan. Sched. 2012. V. 1, No. 3. P. 147–170.

36. Chen W., Shi Y., Teng H., Lan X., Hu L. An efficient hybrid algorithm for resource-constrained project scheduling // Inf. Sci. 2010. V. 180, No. 6. P. 1031–1039.

37. Mendes J. J. M., Goncalves J. F., Resende M. G. C. A random key based genetic algorithm for the resource-constrained project scheduling problem // Comput. Oper. Res. 2009. V. 36. P. 92–109.

38. Debels D., Vanhoucke M. Decomposition-based genetic algorithm for the resource-constrained project scheduling problem // Oper. Res. 2007. V. 55. P. 457–469.

39. Koulinas G., Kotsikas L., Anagnostopoulos K. A particle swarm optimization based hyper-heuristic algorithm for the classic resource constrained project scheduling problem // Inf. Sci. 2014. V. 277. P. 680–693.

40. Valls V., Ballestin F., Quintanilla S. A hybrid genetic algorithm for the resource-constrained project scheduling problem // Eur. J. Oper. Res. 2008. V. 185, No. 2. P. 495–508.

41. Carlier J., Moukrim A., Xu H. A memetic algorithm for the resource constrained project scheduling problem // Proc. 2009 Int. Conf. Ind. Eng. Syst. Manage. (Montreal, Canada, May 13–15, 2009). Piscataway: IEEE, 2009.

42. Debels D., De Reyck Leus B. R., Vanhoucke M. A hybrid scatter search electromagnetism meta-heuristic for project scheduling // Eur. J. Oper. Res. 2006. V. 169, No. 2. P. 638–653.

43. Ranjbar M., Kianfar F. A hybrid scatter search for the RCPSP // Sci. Iran. 2009. V. 16, No. 1. P. 11–18.

44. Jedrzejowicz P., Ratajczak E. Population learning algorithm for the resource-constrained project scheduling // Perspectives in Modern Project Scheduling. New York: Springer, 2006. P. 275–296.

45. Ying K. C., Lin S. W., Lee Z. J. Hybrid-directional planning: Improving improvement heuristics for scheduling resource-constrained projects // Int. J. Adv. Manuf. Technol. 2009. V. 41, No. 3–4. P. 358–366.

46. Alcaraz J., Maroto C. A hybrid genetic algorithm based on intelligent encoding for project scheduling // Perspectives in Modern Project Scheduling. New York: Springer, 2006. P. 249–274.

47. Valls V., Ballestin F., Quintanilla M. S. Justification and RCPSP: A technique that pays // Eur. J. Oper. Res. 2005. V. 165, No. 2. P. 375–386.

Гончаров Евгений Николаевич

Статья поступила
21 апреля 2022 г.
После доработки —
25 мая 2022 г.
Принята к публикации
26 мая 2022 г.

DISKRETNYI ANALIZ I ISSLEDOVANIE OPERATSII
/DISCRETE ANALYSIS AND OPERATIONS RESEARCH/
October–December 2022. Vol. 29, No. 4. P. 15–37

UDC 519.8+518.25 DOI 10.33048/daio.2022.29.734

LOCAL SEARCH ALGORITHM FOR THE RESOURCE-CONSTRAINED PROJECT SCHEDULING PROBLEM

E. N. Goncharov[1,2]

[1] Sobolev Institute of Mathematics,
4 Akad. Koptyug Avenue, 630090 Novosibirsk, Russia
[2] Novosibirsk State University,
2 Pirogov Street, 630090 Novosibirsk, Russia
E-mail: gon@math.nsc.ru

Abstract. We consider the resource-constrained project scheduling problem (RCPSP). The problem accounts for technological constraints of activities precedence together with resource constraints. All resources are renewable. Activities preemptions are not allowed. This problem is NP-hard in the strong sense. We present a new local search algorithm that uses a Tabu-list and two type of neighborhoods. The algorithm is evaluated using three data bases of instances of the problem: 480 instances of 60 activities, 480 of 90, and 600 of 120 activities respectively, taken from the PSPLIB library available online. Numerical experiments demonstrate that the proposed algorithm produces better results than existing algorithms in the literature for large-sized instances. For some instances from the dataset j120, the best known heuristic solutions were improved. Tab. 4, bibliogr. 47.

Keywords: resource-constrained project scheduling problem, renewable resources, Tabu search, variable neighborhood search, PSPLIB.

REFERENCES

1. **P. Brucker, A. Drexl, R. Möhring, K. Neumann,** and **E. Pesch,** Resource-constrained project scheduling: Notation, classification, models, and methods, *Eur. J. Oper. Res.* **112** (1), 3–41 (1999).

The study is carried out within the framework of the state contract of the Sobolev Institute of Mathematics (Project FWNF–2022–0019).

English version: Journal of Applied and Industrial Mathematics **16** (4) (2022).

2. **J. Blazewicz, J. K. Lenstra,** and **A. H. G. Rinnoy Kan,** Scheduling subject to resource constraints: Classification and complexity, *Discrete Appl. Math.* **5** (1), 11–24 (1983).
3. **É. Kh. Gimadi,** On some mathematical models and methods for planning large-scale projects, in *Models and Optimization Methods* (Tr. Inst. Mat., Vol. 10) (Nauka, Novosibirsk, 1988), pp. 89–115.
4. **É. Kh. Gimadi, V. V. Zalyubovskii,** and **S. V. Sevastyanov,** Polynomial solvability of scheduling problems with storable resources and deadlines, *Diskretn. Anal. Issled. Oper., Ser. 2,* **7** (1), 9–34 (2000).
5. **R. Pellerin, N. Perrier,** and **F. Berthaut,** LSSPER: A survey of hybrid metaheuristics for the resource-constrained project scheduling problem, *Eur. J. Oper. Res.* **280** (2), 395–416 (2020).
6. **M. Abdolshah,** A review of resource-constrained project scheduling problems (RCPSP) approaches and solutions, *Int. Trans. J. Eng. Manage. Appl. Sci. Technol.* **5** (4), 253–286 (2014).
7. **M. Vanhoucke,** Resource-constrained project scheduling, in *Project Management with Dynamic Scheduling* (Springer, Heidelberg, 2012), pp. 107–137.
8. **S. Hartmann** and **D. Briskorn,** A survey of variants and extensions of the resource-constrained project scheduling problem, *Eur. J. Oper. Res.* **207** (1), 1–14 (2010).
9. **R. Kolisch** and **S. Hartmann,** Experimental investigation of heuristics for resource-constrained project scheduling: An update, *Eur. J. Oper. Res.* **174** (1), 23–37 (2006).
10. **M. Palpant, C. Artigues,** and **P. Michelon,** Solving the resource-constrained project scheduling problem with large neighborhood search, *Ann. Oper. Res.* **131**, 237–257 (2004).
11. **Yu. A. Kochetov** and **A. A. Stolyar,** Evolutionary local search with variable neighborhood for the resource constrained project scheduling problem, in *Proc. 5th Int. Workshop Comput. Sci. Inf. Technol., Ufa, Russia, Sept. 16–18* (USATU, Ufa, 2003), pp. 96–99.
12. **E. N. Goncharov,** Variable neighborhood search for the resource constrained project scheduling problem, in *Mathematical Optimization Theory and Operations Research* (Rev. Sel. Pap. 18th Int. Conf., Yekaterinburg, Russia, July 8–12, 2019) (Springer, Cham, 2021), pp. 39–50 (Commun. Comput. Inf. Sci., Vol. 1090).
13. **R. Kolisch** and **S. Hartmann,** Heuristic algorithms for solving the resource-constrained project scheduling problem: Classification and computational analysis, in *Project Scheduling: Recent Models, Algorithms and Applications* (Kluwer Acad. Publ., Boston, 1999), pp. 147–178.
14. **E. Kh. Gimadi, E. N. Goncharov,** and **D. V. Mishin,** On some implementations of solving the resource-constrained project scheduling problem, *Yugoslav. J. Oper. Res.* **29** (1), 31–42 (2019).
15. **M. Gagnon, F. F. Boctor,** and **G. d'Avignon,** A tabu search algorithm for the resource-constrained project scheduling problem, in *Proc. 2004 Conf. Adm. Sci. Assoc. Canada, Quebec, Canada, June 5–8, 2004* (ASAC, Waterloo, 2004).

16. **S. Proon** and **M. Jin,** A genetic algorithm with neighborhood search for the resource-constrained project scheduling problem, *Naval Res. Logist.* **58**, 73–82 (2011).
17. **V. Valls, F. Ballestin,** and **S. Quintanilla,** A population-based approach to the resource-constrained project scheduling problem, *Ann. Oper. Res.* **131**, 305–324 (2004).
18. **R. Kolisch** and **A. Sprecher,** PSPLIB — a Project Scheduling Problem Library, *Eur. J. Oper. Res.* **96** (1), 205–216 (1996).
19. *Project Scheduling: Recent Models, Algorithms and Applications* (Kluwer Acad. Publ., Boston, 1999).
20. **R. Kolisch, A. Sprecher,** and **A. Drexl,** Characterization and generation of a general class of resource-constrained project scheduling problems, *Manage. Sci.* **41** (10), 1693–1703 (1995).
21. **E. N. Goncharov** and **V. V. Leonov,** Genetic algorithm for the resource-constrained project scheduling problem *Avtom. Telemekh.,* No. 6, 173–189 (2017) [Russian] [*Autom. Remote Control.* **78** (6), 1101–1114 (2017)].
22. **F. Berthaut, R. Pellerin, A. Hajji,** and **N. Perrier,** A path relinking-based scatter search for the resource-constrained project scheduling problem, *Int. J. Proj. Organ. Manage.* **10** (1), 1–36 (2018).
23. **D. C. Paraskevopoulos, C. D. Tarantilis,** and **G. Ioannou,** Solving project scheduling problems with resource constraints via an event list-based evolutionary algorithm, *Expert Syst. Appl.* **39** (4), 3983–3994 (2012).
24. **M. Mobini, Z. Mobini,** and **M. Rabbani,** An artificial immune algorithm for the project scheduling problem under resource constraints, *Appl. Soft Comput.* **11** (2), 1975–1982 (2011).
25. **M. D. Mahdi Mobini, M. Rabbani, M. S. Amalnik, J. Razmi,** and **A. R. Rahimi-Vahed,** Using an enhanced scatter search algorithm for a resource-constrained project scheduling problem, *Soft Comput.* **13**, 597–610 (2009).
26. **H. Wang, T. Li,** and **T. Lin,** Efficient genetic algorithm for resource-constrained project scheduling problem, *Trans. Tianjin Univ.* **16** (5), 376–382 (2010).
27. **J. Goncalves, M. G. Resende, C,** and **J. Mendes,** A biased random key genetic algorithm with forward-backward improvement for resource-constrained project scheduling problem, *J. Heuristics* **17**, 467–486 (2011).
28. **S. Elsayed, R. Sarker, T. Ray,** and **C. C. Coello,** Consolidated optimization algorithm for resource-constrained project scheduling problems, *Inf. Sci.* **418–419**, 346–362 (2017).
29. **J. Czogalla** and **A. Fink,** Particle swarm topologies for resource constrained project scheduling, in *Nature Inspired Cooperative Strategies for Optimization (NICSO 2008)* (Springer, Heidelberg, 2009), pp. 61–73.
30. **A. Lim, H. Ma, B. Rodrigues, S. T. Tan,** and **F. Xiao,** New metaheuristics for the resource-constrained project scheduling problem, *Flex. Services Manuf. J.* **25** (1–2), 48–73 (2013).

31. D. Chen, S. Liu, and **S. Qin,** Memetic algorithm for the resource-constrained project scheduling problem, in *Proc. 11th World Congr. Intell. Control Autom. (WCICA 2014), Shenyang, China, June 27–30, 2014* (IEEE, Piscataway, 2014), pp. 4991–4996.

32. X. Zheng and **L. Wang,** A multi-agent optimization algorithm for resource constrained project scheduling problem, *Expert Syst. Appl.* **42** (15–16), 6039–6049 (2015).

33. R. Zamani, A competitive magnet-based genetic algorithm for solving the resource-constrained project scheduling problem, *Eur. J. Oper. Res.* **229** (2), 552–559 (2013).

34. X. Zheng and **L. Wang,** An effective shuffled frog-leaping algorithm for resource-constrained project scheduling problem, *Comput. Oper. Res.* **39** (5), 890–901 (2012).

35. I. Y. Ismail and **M. A. Barghash,** Diversity guided genetic algorithm to solve the resource constrained project scheduling problem, *Int. J. Plan. Sched.* **1** (3), 147–170 (2012).

36. W. Chen, Y. Shi, H. Teng, X. Lan, and **L. Hu,** An efficient hybrid algorithm for resource-constrained project scheduling, *Inf. Sci.* **180** (6), 1031–1039 (2010).

37. J. J. M. Mendes, J. F. Goncalves, and **M. G. C. Resende,** A random key based genetic algorithm for the resource-constrained project scheduling problem, *Comput. Oper. Res.* **36**, 92–109 (2009).

38. D. Debels and **M. Vanhoucke,** Decomposition-based genetic algorithm for the resource-constrained project scheduling problem, *Oper. Res.* **55**, 457–469 (2007).

39. G. Koulinas, L. Kotsikas, and **K. Anagnostopoulos,** A particle swarm optimization based hyper-heuristic algorithm for the classic resource constrained project scheduling problem, *Inf. Sci.* **277**, 680–693 (2014).

40. V. Valls, F. Ballestin, and **S. Quintanilla,** A hybrid genetic algorithm for the resource-constrained project scheduling problem, *Eur. J. Oper. Res.* **185** (2), 495–508 (2008).

41. J. Carlier, A. Moukrim, and **H. Xu,** A memetic algorithm for the resource constrained project scheduling problem, in *Proc. 2009 Int. Conf. Ind. Eng. Syst. Manage., Montreal, Canada, May 13–15, 2009* (IEEE, Piscataway, 2009).

42. D. Debels, B. R. De Reyck Leus, and **M. Vanhoucke,** A hybrid scatter search electromagnetism meta-heuristic for project scheduling, *Eur. J. Oper. Res.* **169** (2), 638–653 (2006).

43. M. Ranjbar and **F. Kianfar,** A hybrid scatter search for the RCPSP, *Sci. Iran.* **16** (1), 11–18 (2009).

44. P. Jedrzejowicz and **E. Ratajczak,** Population learning algorithm for the resource-constrained project scheduling, in *Perspectives in Modern Project Scheduling* (Springer, New York, 2006), pp. 275–296.

45. K. C. Ying, S. W. Lin, and **Z. J. Lee,** Hybrid-directional planning: Improving improvement heuristics for scheduling resource-constrained projects, *Int. J. Adv. Manuf. Technol.* **41** (3–4), 358–366 (2009).

46. J. Alcaraz and **C. Maroto,** A hybrid genetic algorithm based on intelligent encoding for project scheduling, in *Perspectives in Modern Project Scheduling* (Springer, New York, 2006), pp. 249–274.

47. V. Valls, F. Ballestin, and **M. S. Quintanilla,** Justification and RCPSP: A technique that pays, *Eur. J. Oper. Res.* **165** (2), 375–386 (2005).

Evgeny N. Goncharov

Received April 21, 2022
Revised May 25, 2022
Accepted May 26, 2022