УДК 519.854.2 + 519.852.2

Д.И. Архипов[1], А.А. Лазарев[1], Г.В. Тарасов[2]

[1]Институт проблем управления им. В.А. Трапезникова РАН, Москва, Россия
[2]Московский государственный университет им. М.В. Ломоносова, Москва, Россия

ОПРЕДЕЛЕНИЕ ЗАГРУЗКИ РЕСУРСОВ ПРИ ПОИСКЕ НИЖНИХ ОЦЕНОК ДЛЯ ЗАДАЧИ RCPSP

Рассмотрена задача определения загрузки ресурсов при поиске нижних оценок общего времени выполнения работ проекта с учетом ограничения на ресурсы (RCPSP). Для случая двух ресурсов предложен геометрический подход к решению задачи и представлены два полиномиальных алгоритма точного решения задачи со сложностью $O(n^2)$ и $O(n \log n)$, где n – исходное число работ.

Ключевые слова: задача составления расписания работ проекта с учетом ограничения на ресурсы, нижняя оценка общего времени выполнения проекта, определение загрузки ресурсов.

D.I. Arkhipov[1], A.A. Lazarev[1], G.V. Tarasov[2]

[1]V.A. Trapeznikov Institute of Control Sciences of the Russian Academy of Sciences, Moscow, Russian Federation
[2]Lomonosov Moscow State University, Moscow, Russian Federation

DETERMINING RESOURCE LOADS FOR ESTIMATING MAKESPAN LOWER BOUNDS FOR RESOURCE-CONSTRAINED PROJECT SCHEDULING PROBLEM

We consider the problem of determining resource loads in search of makespan lower bounds for resource-constrained project scheduling problem (RCPSP). For the case of two resources we propose a geometric approach to solving the problem and present two exact polynomial algorithms with time complexities $O(n^2)$ and $O(n \log n)$, where n is the initial number of jobs.

Keywords: scheduling theory, resource-constrained project scheduling problem, makespan lower bound, determining resource loads.

Введение

Задача построения расписания выполнения работ проекта с учетом ограничений на ресурсы (resource-constrained project scheduling problem, RCPSP) является одной из основных задач теории расписаний [1] и широко применяется в качестве математической модели при управлении циклом производства в масштабных проектах. Задача

RCPSP является *NP*-трудной в сильном смысле [2]. В простейшей формулировке задачи RCPSP проект задается множеством работ и ограничениями доступности ресурсов, требуемых для выполнения работ. Целью является построение расписания с минимальной общей продолжительностью выполнения проекта (makespan).

Большинство точных алгоритмов решения задачи RCPSP основаны на методе ветвей и границ, для применения которого необходимо знание верхних и нижних оценок общей продолжительности проекта. В работе [3] был представлен алгоритм поиска нижних оценок, основанный на определении загрузки ресурсов. Решение задачи нахождения максимальной загрузки заданной пары ресурсов и стало целью данной работы.

Для решения задачи был использован геометрический подход. Работы представляются в виде векторов на плоскости. Показано, что удовлетворяющие заданным ограничениям линейные комбинации этих векторов образуют на плоскости некоторый выпуклый многоугольник, названный областью разрешимости задачи. Показано, что необходимым и достаточным условием существования оптимального плана выполнения работ является условие принадлежности точки (A, B) этому многоугольнику. Представлены два алгоритма решения задачи, имеющие вычислительные сложности $O(n^2)$ и $O(n \log n)$ операций соответственно, где n – исходное число работ. Данные алгоритмы были использованы при проведении численных экспериментов, результаты которых были представлены в работе [3].

1. Формулировка задачи

Имеются ресурсы двух видов в количестве $A, B \in \mathbb{R}^+$ единиц и множество $N = \{1, ..., n\}$ работ. На выполнение работы $i \in N$ в полном объеме расходуется a_i, b_i единиц ресурсов. Работы могут выполняться частично: выполняемая часть i-й работы может составлять $0 \le x_i \le 1$ от ее полного объема, при этом будет израсходовано $a_i x_i$, $b_i x_i$ единиц ресурсов соответственно. Требуется определить загрузки ресурсов – составить план выполнения работ, минимизирующий совокупный объем неизрасходованных ресурсов:

$$\left(\left(A - \sum_{i=1}^{n} a_i x_i \right)^2 + \left(B - \sum_{i=1}^{n} b_i x_i \right)^2 \right)^{1/2} \to \min,$$

а загрузки ресурсов A и B (максимальный возможный их расход) равны $\sum_{i=1}^{n} a_i x_i$ и $\sum_{i=1}^{n} b_i x_i$ в соответствии с данным оптимальным планом.

2. Сопутствующие задачи

Задача о сумме подмножества. Имеется множество $S = \{s_i\}$ действительных чисел мощностью n. Необходимо определить, существует ли для заданного действительного числа s непустое подмножество $S' = \{s_i\}$, $S' \subset S$, $\| S \| = m \le n$, имеющее заданную сумму $\sum_{i=1}^{m} s_i = s$. Эквивалентная постановка: определить, существует ли булев вектор $\vec{x} | \forall i \in \overline{1,n} \quad x_i \in \{0,1\}$, такой что $\sum_{i=1}^{n} x_i s_i = s$. Эта задача является *NP*-трудной и может рассматриваться как частный случай задачи о рюкзаке без повторений ({0 – 1} knapsack problem), для которой существует алгоритм с псевдополиномиальной вычислительной сложностью [5].

Задача о сумме подмножества с дроблениями. Имеется множество $S = \{s_i\}$ действительных чисел мощностью n. Требуется определить, существует ли вектор $\vec{x}$ с ограниченными вещественными компонентами $0 \le x_i \le 1$, такой что $\sum_{i=1}^{n} x_i s_i = s$. Благодаря тому что компоненты вектора-решения более не являются строго булевыми, данная задача, в отличие от предыдущей, не является *NP*-трудной: очевидно, пока текущее значение суммы искомой линейной комбинации $\sum_{i=1}^{k} x_i s_i$ и очередного числа s_{k+1} меньше целевого значения s, можно добавлять очередное число полностью ($x_{k+1} = 1$), в противном случае $x_{k+1} = \left(s - \sum_{i=1}^{k} x_i s_i \right) / s_{k+1}$. Таким образом, задача разрешима за время $O(n)$.

Задача суммирования векторов. Имеется множество n векторов $S = \{s_i\}$ в векторном пространстве размерности k. Вводится метрика $\|.\|$ на данном векторном пространстве (например, евклидова метрика l_2). Требуется найти булев вектор $X = \{x_i\}$, $x_i \in \{0,1\}$, такой что метрика $\| S' \|$ вектора $S' = \sum_{i=1}^{n} x_i s_i$ достигает максимума. В работе [4] приводятся полиномиальные алгоритмы решения задачи суммирования векторов для случая произвольной размерности для евклидовой и полиэдральной метрик. Однако, в отличие от задачи, рассмотренной в данной

работе, исследованная в статье задача не содержит ограничений на значения координат искомой линейной комбинации («ограничений на ресурсы»), а искомый вектор является строго булевым.

3. Геометрический подход

Сформулируем постановку задачи в виде задачи нелинейного программирования.

Дано: пара чисел A, $B \in \mathbb{R}^+$, множество векторов на плоскости $S = \{s_i\}$, $s_i = (a_i, b_i)$ с неотрицательными вещественными координатами $a_i, b_i \in \mathbb{R}^+$, $i \in \overline{1,n}$.

Найти: вектор $X = \{x_i\}$ с ограниченными вещественными компонентами $x_i \in [0,1]$, $i \in \overline{1,n}$, такой что $A \geq \sum_{i=1}^{n} a_i x_i$, $B \geq \sum_{i=1}^{n} b_i x_i$ и $\left(\left(A - \sum_{i=1}^{n} a_i x_i\right)^2 + \left(B - \sum_{i=1}^{n} b_i x_i\right)^2\right) \to \min$.

3.1. Исследование математическом модели. Условие разрешимости

Рассмотрим перестановки γ_{up} и γ_{down} из n векторов из множества S, получаемые сортировкой элементов этого множества по неубыванию и по невозрастанию соотношения их координат b_i/a_i. Для последовательности γ_{up}

$$\forall i \in \overline{1,n-1} \quad \frac{b_i^{\gamma_{up}}}{a_i^{\gamma_{up}}} \geq \frac{b_{i+1}^{\gamma_{up}}}{a_{i+1}^{\gamma_{up}}}.$$

Для перестановки γ_{down} неравенство обратное. Операция деления здесь несколько «формальная»: случай деления на ноль стоит обрабатывать отдельным условным оператором. В случае, когда у каких-либо двух векторов совпадают соотношения b_i/a_i их координат, порядок следования этих векторов считаем несущественным.

Отложим первый вектор из γ_{up} от начала координат на плоскости, второй – от конца первого и т.д. Получим некоторую ломаную (обозначим ее также γ_{up}) на плоскости. Обозначим как θ_i величину угла между вектором $\vec{s}_i$ и осью абсцисс a соответственно. Поскольку сортировки производятся по величине $\mathrm{tg}\theta_i$, γ_{up} является выпуклой вверх

в каждой своей точке излома. Аналогично получим ломаную γ_{down}, выпуклую вниз в каждой своей точке излома.

Назовем областью разрешимости задачи для множества *S* плоский многоугольник, образованный ломаными γ_{up}, γ_{down}, отложенными от начала координат на плоскости. Очевидно, этот плоский многоугольник является выпуклым.

Теорема 1. Для любого вектора $\vec{x}$, компоненты которого x_i удовлетворяют ограничениям $0 \le x_i \le 1$ (при этом $\vec{x}$ необязательно соответствует решению задачи), точка $\left(\sum_{i=1}^{n} a_i x_i, \sum_{i=1}^{n} b_i x_i\right)$, $s_i = (a_i, b_i)$, $s_i \in S$ лежит в пределах (внутри или на границе) области разрешимости множества *S*.

Доказательство. Построим на плоскости $b(a)$ некоторую ломаную γ из векторов s_i^{γ} множества *S*, такую что каждая ее точка лежит не ниже любой ломаной ψ, также состоящей из векторов множества *S*:

$$b^{\gamma}(a) \ge b^{\psi}(a),$$

где $b^{\gamma}(a)$, $b^{\psi}(a)$ – кусочно-заданные функции – уравнения ломаных γ, ψ на координатной плоскости $b(a)$. Докажем, что ломаная γ совпадает с вышеупомянутой ломаной γ_{up}, т.е. является выпуклой вверх в каждой своей точке излома. Предположим обратное: пусть γ не является выпуклой вверх в *i*-й точке излома, т.е.

$$\exists i \in \overline{1, n-1} \quad \frac{b_i^{\gamma}}{a_i^{\gamma}} < \frac{b_{i+1}^{\gamma}}{a_{i+1}^{\gamma}}.$$

Тогда составим ломаную γ', совпадающую с γ, за исключением того, что *i* и *i*+1 векторы из γ в ней следуют в обратном порядке: $s_i^{\gamma} = s_{i+1}^{\gamma'}$ $s_{i+1}^{\gamma} = s_i^{\gamma'}$. При этом γ' является выпуклой вверх в *i*-й точке излома, и, очевидно, на отрезке $\left(\sum_{j=1}^{i} a_j^{\gamma}, \sum_{j=1}^{i+1} a_j^{\gamma}\right)$ γ' будет лежать выше γ: $b^{\gamma}(a) < b^{\gamma'}(a)$, что противоречит условию теоремы. Значит, γ совпадает с γ_{up}. Аналогично доказывается, что для произвольной ломаной γ' $b^{\gamma'}(a) \ge b^{\gamma_{down}}(a)$. Очевидно, любая ломаная ψ, полученная из некоторой перестановки векторов $x_i \cdot \vec{s}_i$, $0 \le x_i \le 1$, а значит, и точка

$\left(\sum_{i=1}^{n} a_i x_i, \sum_{i=1}^{n} b_i x_i\right)$ также будет лежать не выше ломаной γ_{up} и не ниже ломаной γ_{down}, т.е. в пределах многоугольника, образованного этими ломаными – в пределах области разрешимости. Таким образом, теорема доказана.

Следствие 1 (необходимое и достаточное условие разрешимости задачи). Для того чтобы задача была разрешима, необходимо и достаточно, чтобы точка (A, B) лежала в пределах области разрешимости множества S.

Таким образом, мы получили способ проверить разрешимость задачи для конкретных входных данных: для этого надо отсортировать векторы из заданного множества S по соотношению их координат b_i/a_i, построить на плоскости область разрешимости множества S и проверить, содержит ли она точку (A, B).

4. Поиск решения. Построение алгоритмов

4.1. Алгоритм 1

Рассмотрим какое-либо фиксированное множество работ S и вектор ресурсов (A, B). Пусть соответствующая задача имеет частное решение $\vec{x} = \{x_1, x_2, ..., x_n\}$. Заметим, что если $\exists i \in \overline{1,n} : x_i = 0$ (или 1), то решение $\vec{x}$ такой задачи совпадает с решением $\vec{x'}$ задачи для множества работ $S' = S/s_i$ с вектором ресурсов (A, B) (или ($A - a_i$, $B - b_i$) соответственно) с той лишь разницей, что в $\vec{x'}$ «пропущена» компонента x_i.

Таким образом, мы получаем стратегию, в соответствии с которой будем понижать множество имеющихся работ и переходить к задачам с меньшим числом работ, пока оно не станет достаточно малым (0, 1 или 2), чтобы задача свелась к решению простейшей системы линейных алгебраических уравнений.

Алгоритм 1:

Шаг 1. Отсортировать работы исходного множества работ S по невозрастанию соотношения b_i/a_i.

Шаг 2. Проверить разрешимость задачи для множества S (проверить, принадлежит ли точка (A, B) области разрешимости для множества работ S), если задача неразрешима – конец алгоритма.

Шаг 3. Положить $i = 1$.

Цикл 1:

а) для очередной работы $s_i \in S$ построить подмножество работ $S_i' = S \setminus s_i$;

б) проверить, является ли разрешимой задача для множества работ S_i' (см. шаг 2); если является, в ответ записать $x_i = 0$, положить $S := S_i'$ (с сохранением порядка элементов в множестве) и повторить цикл для следующей работы (перейти к началу шага 3), если таковая имеется.

Шаг 4. Положить $i = 1$.

Цикл 2:

а) для каждой работы $s_i \in S$ построить подмножество работ $S_i' = S \setminus s_i$ и модифицированный вектор ресурсов $A' = A - a_i$, $B' = B - b_i$ (примечание: здесь и далее A, B не исходные количества ресурсов, а их текущие значения, полученные в процессе работы алгоритма);

б) проверить, является ли разрешимой задача для множества S_i' и пары ресурсов A', B'; если является, в ответ записать $x_i = 1$, положить $S := S_i'$, $A := A'$, $B := B'$ и повторить цикл для следующей работы, если таковая имеется.

Шаг 5. Если множество S содержит две работы j, k, решить систему уравнений

$$\begin{cases} x_j a_j + x_k a_k = A, \\ x_j b_j + x_k b_k = B. \end{cases}$$

Если S содержит 1 или 0 работ, решение очевидно.

Шаг 6. Конец работы алгоритма.

Утверждение 1. Алгоритм 1 корректен, т.е. имеет конечное время работы, и результатом работы алгоритма является решение задачи в случае существования такового.

Справедливость этого утверждения доказывается путем простого рассмотрения структуры алгоритма.

Утверждение 2. Алгоритм 1 имеет вычислительную сложность $O(n^2)$ операций, где n – исходное число работ.

Для доказательства достаточно последовательно рассмотреть вычислительную сложность каждого шага алгоритма.

4.2. Алгоритм 2

Рассмотрим исходное множество работ S. Построим на координатной плоскости ранее описанным способом соответствующий этому множеству работ многоугольник (область разрешимости), обозначим его M. Построим на той же координатной плоскости многоугольник M', полученный параллельным переносом M на вектор (A, B). Обозначим его M'. Поскольку M и M' – выпуклые многоугольники и переходят друг в друга при параллельным переносе на вектор $\pm\overrightarrow{(A+B)}$ соответственно, то они могут либо пересекаться не более чем в двух точках либо иметь общую грань или ее участок. Рассмотрим одну из точек пересечения (на рисунке она обозначена как C).

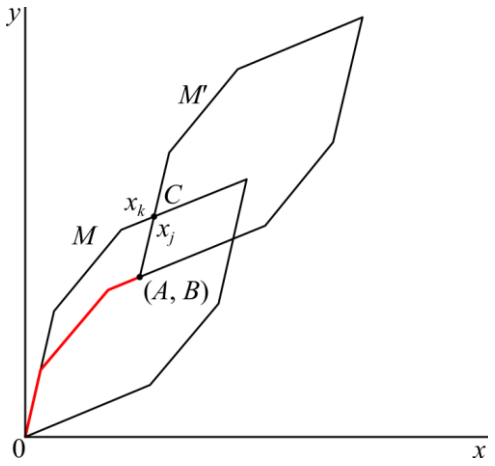

Рис. Пояснение к алгоритму 2

Для определения индексов пересекающихся звеньев j, k необходимо для всех пар $j,k \in \overline{1,n}$ решить систему

$$\begin{cases} \sum_{i<k} a_i + x_k a_k = A + \sum_{i<j} a_i + x_j a_j, \\ \sum_{i<k} b_i + x_k b_k = B + \sum_{i<j} b_i + x_j b_j \end{cases} \quad (1)$$

относительно x_j, x_k и проверить, удовлетворяет ли полученное решение условию $0 \le x_j, x_k \le 1$.

Для координат x_C, y_C точки C и коэффициентов x_j, x_k имеем следующую систему уравнений:

$$\begin{cases} \sum_{i<k} a_i + x_k a_k = x_C, \\ \sum_{i<k} b_i + x_k b_k = y_C, \\ A + \sum_{i<j} a_i + x_j a_j = x_C, \\ B + \sum_{i<j} b_i + x_j b_j = y_C. \end{cases} \quad (2)$$

Заметим, что в силу того, что $A, B > 0$ и $\forall i \in \overline{1,n}$ $a_i, b_i > 0$, имеем $j \le k$. Исключаем из этой системы x_C, y_C и переносим суммы в левую часть, получаем

$$\begin{cases} \sum_{i<k} a_i + x_k a_k - \sum_{i<j} a_i - x_j a_j = A, \\ \sum_{i<k} b_i + x_k b_k - \sum_{i<j} b_i - x_j b_j = B. \end{cases} \quad (3)$$

Преобразуем:

$$\begin{cases} (1 - x_j) a_j + \sum_{j<i<k} a_i + x_k a_k = A, \\ (1 - x_j) b_j + \sum_{j<i<k} b_i + x_k b_k = B. \end{cases} \quad (4)$$

Обозначим $\alpha = 1 - x_j$, $\beta = x_k$ (здесь x_j, x_k – найденное ранее решение системы (2), удовлетворяющее условию $0 \le x_j, x_k \le 1$). Таким образом, пересекающиеся звенья j, k многоугольников M', M соответствуют работам j, k, входящим в решение с коэффициентами $0 < \alpha, \beta < 1$.

Тогда можно построить следующий алгоритм.

Алгоритм 2:

Шаг 1. Отсортировать работы исходного множества работ S по невозрастанию соотношения b_i/a_i.

Шаг 2. Проверить разрешимость задачи для множества S (проверить, принадлежит ли точка (A, B) области разрешимости для множества работ S); если задача неразрешима, конец алгоритма.

Шаг 3. Положить $j = 1$.

Цикл:

Методом дихотомии определить, существует ли такое $k \in \overline{1,n}$, что решение системы (3) относительно x_j, x_k удовлетворяет условию $0 \le x_k \le 1$. Если такого k не существует, повторить цикл для следующей работы $(j + 1)$.

Если такое k существует, но при этом не выполнено условие $0 \le x_j \le 1$, повторить цикл для следующей работы $(j + 1)$.

Если такое k существует и при этом выполнено условие $0 \le x_j \le 1$, положить $\alpha = 1 - x_j$, $\beta = x_k$, завершить цикл.

Шаг 4. Положить:

$$\forall i < j : x_i = 0,$$

$$x_j = \alpha,$$

$$\forall i \mid j < i < k : x_i = 1,$$

$$\forall i > k : x_i = 0.$$

Шаг 5. Конец работы алгоритма.

Утверждение 3. Алгоритм 2 корректен, т.е. имеет конечное время работы и результатом работы алгоритма является решение задачи в случае существования такового.

Справедливость этого утверждения доказывается путем простого рассмотрения структуры алгоритма.

Утверждение 4. Алгоритм 2 имеет вычислительную сложность $O(n \log n)$ операций, где n – исходное число работ.

Для доказательства достаточно последовательно рассмотреть вычислительную сложность каждого шага алгоритма.

Таким образом, алгоритм 1 проигрывает в вычислительной сложности и, более того, является достаточно громоздким при реализации, однако он допускает модификации для решения аналогичных задач (например, если требуется, помимо прочего, максимизировать число ненулевых компонент в ответе или если работы имеют «вес» и требуется, помимо прочего, максимизировать совокупный вес входящих в решение работ), в то время как алгоритм 2 такой особенностью не обладает.

Заключение

Представленные в данной работе алгоритмы решения учитывают все необходимые условия и ограничения, входящие в постановку задачи. Поскольку данные алгоритмы предоставляют точное решение и обладают полиномиальным временем работы, представляется возможным их использование на практике. Алгоритм 2 был использован в работе [3] в качестве составной части алгоритма решения задачи поиска нижней оценки общего времени выполнения работ проекта с ограниченными ресурсами. Представленный в работе [3] алгоритм поиска нижней оценки имеет вычислительную сложность $O(n^3 r(n + m + r)H \times \times \log H)$, где n – количество работ, r – количество ресурсов, H – гори-

зонт планирования, m – количество точек излома у функций емкости ресурсов. Наличие компоненты $O(n^3)$ обусловлено иной составной частью алгоритма, не содержащей алгоритм 2.

В дальнейшем планируется развитие разработанного подхода для случая произвольного числа ресурсов для задач с дополнительными требованиями к входящим в решение работам (задача с весами, задача с ограничением на число небулевых компонент решения и т.д.), а также применение разработанных алгоритмов в задаче об инвестициях.

Список литературы

1. Лазарев А.А., Гафаров Е.Р. Теория расписаний. Задачи и алгоритмы. – М.: Физический факультет МГУ, 2011.

2. Garey M.R., Johnson D.S. Complexity results for multiprocessor scheduling under resource constraints // SIAM Journal on Computing. – 1975. – № 4 (4). – P. 397–411.

3. Arkhipov D., Battaia O., Cegarra J. Resource load-based makespan estimation algorithm for resource-constrained project scheduling problem // Proceedings of ROADEF 2017 Conference. – Metz, France, 2017.

4. Бабурин А.Е., Пяткин А.В. О полиномиальных алгоритмах решения одной задачи суммирования векторов // Дискретный анализ и исследование операций. – 2006. – Т. 13 (2). – С. 3–10.

5. Pisinger D. A minimal algorithm for the 0–1 knapsack problem // Operations Research. – 1997. – Vol. 45, № 5. – P. 758–767.

References

1. Lazarev A.A., Gafarov E.R. Teoriia raspisanii. Zadachi i algoritmy [The scheduling theory. Problems and algorithms] Moscow, Fizicheskii fakul'tet MGU. 2011.

2. Garey M.R., Johnson D.S. Complexity results for multiprocessor scheduling under resource constraints – SIAM Journal on Computing. 1975, vol. 4, iss. 4, pp. 397-411.

3. Arkhipov D., Battaia O., Cegarra J. Resource load-based makespan estimation algorithm for resource- constrained project scheduling problem – Proceedings of ROADEF 2017 Conference. Metz, France, 2017.

4. Baburin A.E., Piatkin A.V. O polinomial'nykh algoritmakh resheniia odnoi zadachi summirovaniia vektorov [Polynomial algorithms for the solution

of the problem of summation of vectors] – Diskretnyi analiz i issledovanie operatsii. 2006, vol. 13, iss. 2, pp. 3-10.

5. Pisinger D. A minimal algorithm for the 0–1 knapsack problem – Operations Research. 1997, vol. 45, no. 5, pp. 758-767.

Получено 20.06.2017

Об авторах

Архипов Дмитрий Игоревич (Москва, Россия) – научный сотрудник лаборатории № 68 «Теории расписаний и дискретной оптимизации», Институт проблем управления им. В.А. Трапезникова РАН (117997, г. Москва, ул. Профсоюзная, 65, e-mail: miptrafter@gmail.com).

Лазарев Александр Алексеевич (Москва, Россия) – доктор физико-математических наук, профессор, заведующий лабораторией № 68 «Теории расписаний и дискретной оптимизации», Институт проблем управления им. В.А. Трапезникова РАН (117997, г. Москва, ул. Профсоюзная, 65, e-mail: jobmath@mail.ru).

Тарасов Герман Владимирович (Москва, Россия) – студент, Московский государственный университет им. М.В. Ломоносова (119991, г. Москва, мкр. Ленинские горы, 1, e-mail: gv.tarasov@physics.msu.ru).

About the authors

Dmitrii I. Arkhipov (Moscow, Russian Federation) – Researcher Fellow, Laboratory No. 68 “The Scheduling Theory and Discrete Optimization”, V.A. Trapeznikov Institute of Control Sciences of the Russian Academy of Sciences (65, Profsoyuznaya st., Moscow, 117997, Russian Federation, e-mail: miptrafter@gmail.com).

Aleksandr A. Lazarev (Moscow, Russian Federation) – Doctor of Physics and Mathematics, Professor, Head of the Laboratory No. 68 “The Scheduling Theory and Discrete Optimization”, V.A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences (65, Profsoyuznaya st., Moscow, 117997, Russian Federation, e-mail: jobmath@mail.ru).

German V. Tarasov (Moscow, Russian Federation) – Student, Lomonosov Moscow State University (1, Leninskie Gory, Moscow, 119991, Russian Federation, e-mail: gv.tarasov@physics.msu.ru).