Серия «Математика»
2026. Т. 56. С. 3—18

Онлайн-доступ к журналу:
http://mathizv.isu.ru

ИЗВЕСТИЯ
Иркутского
государственного
университета

Научная статья

УДК 519.8
MSC 90C27, 90C59, 68W25
DOI https://doi.org/10.26516/1997-7670.2026.56.3

Решение задачи минимизации максимального временного смещения при наличии выпуклого ограничения по ресурсу

Ю. В. Захарова[1,2], Д. Д. Городецкий[2], А. О. Захаров[1]✉

[1] Омский филиал Института математики им. С. Л. Соболева СО РАН, Омск, Российская Федерация
[2] Омский государственный университет им. Ф. М. Достоевского, Омск, Российская Федерация
✉ azakharov@ofim.oscsbras.ru

Аннотация: Требуется составить расписание выполнения работ на машинах. Каждая работа характеризуется объемом и числом используемых машин. Минимизируется максимальное временное смещение от заданных директивных сроков работ. Длительности работ зависят от потребления ресурса через выпуклую функцию. Общий доступный объем ресурса ограничен. Для задачи исследуется ее алгоритмическая сложность и предлагаются методы приближенного решения в частных случаях. Строятся модели математического программирования, позволяющие получить новые свойства расписаний.

Ключевые слова: расписание, алгоритм, аппроксимация, ресурс

Благодарности: Исследование выполнено за счет гранта Российского научного фонда № 22-71-10015-П.

Ссылка для цитирования: Захарова Ю. В., Городецкий Д. Д., Захаров А. О. Решение задачи минимизации максимального временного смещения при наличии выпуклого ограничения по ресурсу // Известия Иркутского государственного университета. Серия Математика. 2026. Т. 56. С. 3–18.
https://doi.org/10.26516/1997-7670.2026.56.3

Research article

Solving the Problem of Minimizing the Maximum Lateness with a Convex Resource Constraint

Yulia V. Zakharova[1,2], **Dmitry D. Gorodecky**[2], **Aleksey O. Zakharov**[1]✉

[1] Omsk Branch of the Sobolev Institute of Mathematics SB RAS, Omsk, Russian Federation

[2] Dostoevsky Omsk State University, Omsk, Russian Federation

✉ azakharov@ofim.oscsbras.ru

Abstract: This paper addresses a scheduling problem where each job is characterized by its processing requirements and machine needs. The optimization criterion minimizes maximum lateness relative to given job deadlines, with job durations following a convex resource consumption function under limited total resource availability. We analyze the problem's computational complexity, develop approximation methods for special cases, and construct mathematical programming models that yield new schedule properties.

Keywords: schedule, algorithm, approximation, resource

Acknowledgements: The research was supported by Russian Science Foundation grant N 22-71-10015-П.

For citation: Zakharova Y. V., Gorodetsky D. D., Zakharov A. O. Solving the Problem of Minimizing the Maximum Lateness with a Convex Resource Constraint. *The Bulletin of Irkutsk State University. Series Mathematics*, 2026, vol. 56, pp. 3–18. (in Russian) https://doi.org/10.26516/1997-7670.2026.56.3

1. Введение

Требуется составить расписание выполнения n работ на m машинах. Для каждой работы j, $j \in J = \{1, \ldots, n\}$, задан объем работы W_j, число требуемых машин $size_j$ и директивный срок d_j. Считается, что W_j — это объем работы $j \in J$ на каждой из $size_j$ машин и все задействованные при выполнении работы j машины должны делать ее одновременно. Задача рассматривается как в варианте с прерываниями, так и в варианте без прерываний.

Длительность работы p_j зависит от объема потребляемого ресурса r_j по правилу $p_j(r_j) = \frac{(W_j)^{\kappa+1}}{(r_j)^{\kappa}}$, где $0 < \kappa \leq 1$ — заданная константа. Общий объем ресурса ограничен в совокупности величиной R. В задаче минимизируется максимальное временное смещение $L_{\max} = \max_{j \in J}\{C_j - d_j\}$, где C_j — момент окончания работы j. Будем обозначать через $P|size_j, res|L_{\max}$ ($P|size_j, pmtn, res|L_{\max}$) вариант задачи без прерываний (с прерываниями).

Интенсивность выполнения (скорость) работы j вычисляется как $s_j := \frac{W_j}{p_j} = \frac{W_j (r_j)^{\kappa}}{(W_j)^{\kappa+1}} = \left(\frac{r_j}{W_j}\right)^{\kappa}$.

Интенсивность потребления ресурса работой j на каждой задействованной машине вычисляется как $v_j := \frac{r_j}{p_j} = \frac{r_j (r_j)^\kappa}{(W_j)^{\kappa+1}} = \left(\frac{r_j}{W_j}\right)^{\kappa+1}$. Тогда $v_j = (s_j)^{\frac{\kappa+1}{\kappa}}$, $j \in J$.

С помощью сводимости задачи Разбиение [5] можно показать NP-трудность задачи без прерываний уже при двух машинах и задачи с прерываниями при произвольном числе машин, используя подход из [9].

Задачи, в которых потребление ресурса задается выпуклой функцией, а критерием выступает минимизация длины расписания, рассматривались в [13; 14]. Здесь исследовались варианты задачи с параллельными однородными машинами и системами поточного типа, выделены полиномиально разрешимые случаи и предложены методы решения.

Частным случаем является задача составления расписаний в компьютерных системах, где учитывается расход энергии, определяемый выпуклой функцией [6]. Для различных вариантов этой задачи исследована алгоритмическая сложность и предложены алгоритмы с гарантированной оценкой точности в случае последовательных работ. Одномашинный вариант задачи с учетом расхода энергии и критерием $L_{\max}$, а также агрегированным критерием $L_{\max}$ и расход энергии детально проанализирован в [2], предложены полиномиальные алгоритмы. Разработаны и экспериментально исследованы полиномиальные приближенные алгоритмы для одного или двух процессоров при учете расхода энергии с критерием $L_{\max}$ [1]. Подходы к решению задачи с минимизацией расхода энергии при возможности распараллеливания (модель с конфигурациями работ и метод эллипсоидов, конструктивные двухэтапные алгоритмы) предложены в [9–11].

Полиномиально разрешимые и NP-трудные случаи задачи open-shop с учетом расхода энергии выделены в [15], здесь же рассмотрен агрегированный критерий учета длины расписания и потребления энергии.

В настоящей статье предлагаются методы решения на основе конструктивных алгоритмов, методов математического программирования и эволюционных вычислений для многомашинных вариантов задачи с учетом расхода ресурса, задаваемого выпуклой функцией и критерием минимизации максимального временного смещения.

2. Задача с прерываниями

Дискретное множество скоростей. Предположим, что множество скоростей выполнения работ дискретно и работы упорядочены по неубыванию директивных сроков $d_1 \le d_2 \le \cdots \le d_n$.

Под конфигурацией $c \in C$ будем понимать подмножество работ, которые могут выполняться одновременно с учетом числа используемых машин, совместно с заданными скоростями работ ($s_{j,c}$ — скорость

работы j в конфигурации c). Пусть C_l обозначает подмножество конфигураций, состоящих из работ подмножества $J_l := \{l, l+1, \ldots, n\}$. Введем интервалы $I_l = [d_{l-1} + L,\ d_l + L],\ l = 2, \ldots, n$ и $I_1 = [d_0 := 0, d_1 + L]$. Здесь $L \geq 0$ — вещественная переменная, отвечающая за максимальное временное смещение. Также введем вещественные переменные $x_c^l \geq 0$ — длительность каждой работы конфигурации c в интервале I_l, $l = 1, \ldots, n$. Суммарная интенсивность потребления ресурса в единицу времени в конфигурации $c \in C$ составляет $v_c = \sum_{j \in c} size_j(s_{j,c})^{\frac{\kappa+1}{\kappa}}$.

Модель выпуклого программирования записывается следующим образом:

$$
\begin{gathered}
L \to \min, \\
\textstyle\sum_{c \in C_1} x_c^1 \leq d_1 - d_0 + L, \\
\textstyle\sum_{c \in C_l} x_c^l \leq d_l - d_{l-1},\ l = 2, \ldots, n, \\
\textstyle\sum_{l=1}^{n} \sum_{c \in C_l:\ j \in c} \frac{x_c^l s_{j,c}}{W_j} \geq 1,\ j = 1, \ldots, n, \\
\textstyle\sum_{l=1}^{n} \sum_{c \in C_l} v_c x_c^l \leq R, \\
x_c^l \geq 0,\ L \geq 0,\ l = 1, \ldots, n,\ c \in C_l.
\end{gathered}
$$

Данная задача полиномиально разрешима с помощью метода эллипсоидов с отделяющим оракулом (см., например, [7]). Действительно, в [7] показано, что если двойственная задача линейного программирования разрешима за полиномиальное время, то прямая задача также может быть решена за полиномиальное время. Рассмотрим двойственную задачу линейного программирования:

$$
\begin{gathered}
\textstyle\sum_{l=1}^{n} (d_{l-1} - d_l) y_l + \sum_{j \in \mathcal{J}} z_j - R\mu \to \max, \\
y_1 = 1, \\
\textstyle\sum_{j \in c} \frac{s_{j,c} z_j}{W_j} - y_l - v_c \mu \leq 0,\ c \in C_l,\ l = 1, \ldots, n, \\
z_j \geq 0,\ j \in \mathcal{J}, y_l \geq 0,\ l = 1, \ldots, n, \mu \geq 0.
\end{gathered}
$$

Двойственная задача имеет $O(n)$ переменных и экспоненциальное число ограничений. *Отделяющий оракул* — это алгоритм, который при заданном решении задачи линейного программирования определяет, допустимо это решние или нет, и в случае отрицательного ответа идентифицирует нарушенное ограничение. Далее будут построены полиномиальные отделяющие оракулы для двойственной задачи линейного программирования. Поэтому двойственная задача может быть решена методом эллипсоидов за полиномиальное время (см. [7]). Таким образом, оптимальное решение с заданной точностью $\varepsilon > 0$ может быть найдено для исходной (прямой) задачи за полиномиальное время.

Отделяющий оракул для двойственной задачи линейного программирования работает следующим образом. Для поиска нарушенного ограничения для каждого $l = 1, \ldots, n$ достаточно найти максимум среди

значений $\sum_{j\in c}(\frac{s_{j,c}z_j}{W_j} - size_j(s_{j,c})^{\frac{\kappa+1}{\kappa}}\mu) \le y_l,\ c \in C_l,\ l = 1,\dots,n$, по всем конфигурациям $c \in C_l$. Если это максимальное значение больше (y_l), то имеет место нарушенное ограничение. В противном случае решение двойственной задачи допустимо.

Для каждой работы $j \in J_l$ величина $(\frac{s_{j,c}z_j}{W_j} - size_j(s_{j,c})^{\frac{\kappa+1}{\kappa}}\mu)$ достигает максимального значения при дискретном значении s'_j, которое отыскивается как одна из двух ближайших дискретных скоростей к значению $\left(\frac{z_j\kappa}{(\kappa+1)W_j size_j\mu}\right)^{\kappa}$.

Для работы j дискретное значение s'_j вычисляется с помощью бинарного поиска за время, полиномиальное от длины входа задачи. В итоге необходимо найти конфигурацию $c \in C_l$, при которой максимизируется $\sum_{j\in c}\left(\frac{s'_j z_j}{W_j} - size_j(s'_j)^{\frac{\kappa+1}{\kappa}}\mu\right)$. Задача максимизации этого выражения сводится к одномерной задаче о рюкзаке:

$$\begin{gathered}\textstyle\sum_{j\in\mathcal{J}} b_j y_j \to \max,\\ \textstyle\sum_{j\in\mathcal{J}} a_j y_j \le m,\\ y_j \in \{0,1\},\ j \in \mathcal{J}_l,\end{gathered}$$

где $b_j = \left(\frac{s'_j z_j}{W_j} - size_j(s'_j)^{\frac{\kappa+1}{\kappa}}\mu\right)$ и $a_j = size_j$ для всех $j \in \mathcal{J}_l$. Эта задача может быть решена с помощью алгоритма динамического программирования за время, полиномиальное от m и размера входа задачи.

Лемма 1. *Для задачи $P|size_j, pmtn, discr, res|L_{\max}$ можно построить оптимальное расписание за время, полиномиальное от m и размера входа задачи.*

Непрерывное множество скоростей. В случае непрерывного множества скоростей для работ мощность C не ограничена, поскольку скорость выполнения работы может принимать любые действительные значения. Выполним дискретизацию возможных значений для скорости и рассмотрим только конечное число скоростей, с которыми можно выполнять работы. Таким образом, задача с непрерывным множеством скоростей сводится к задаче с дискретным набором скоростей для работ.

Для дискретизации скоростей найдем нижнюю и верхнюю границу на скорости для работ. Для получения нижней границы S_{LB} предположим, что единственная работа с суммарным объемом $\sum_{j\in\mathcal{J}} W_j size_j$ потребляет весь доступный объем ресурса R, тогда $s_{LB} = \left(\frac{R}{\sum_{j\in J} W_j\cdot size_j}\right)^{\kappa}$. Верхняя граница на скорость равна $s_{UB} = \left(\frac{R}{\min_{j\in J} W_j\cdot size_j}\right)^{\kappa}$, что соответствует случаю назначения всего объема ресурса R работе с наименьшим значением $j' := \operatorname{argmin}_{j\in\mathcal{J}} W_j size_j$.

Рассмотрим указанные нижнюю и верхнюю границы и константу $\delta > 0$. Будем учитывать только скорости из набора $S_\delta = \{S_{LB}, (1 +$

$\delta)S_{LB}, (1+2\delta)S_{LB}, \ldots, (1+k\delta)S_{LB}\}$, где k – наименьшее целое число, такое что $(1+k\delta)S_{LB} \geq S_{UB}$. Следовательно, общее число скоростей ограничено величиной $k \leq \frac{S_{UB}}{\delta S_{LB}}$, что имеет полиномиальную зависимость от $\frac{1}{\delta}$ и экспоненциальную зависимость от размера входа.

Положим δ равным $\left(1+\frac{\theta}{R}\right)^{\kappa} - 1$. Далее рассмотрим оптимальное расписание и округлим скорости работ до ближайших больших дискретных значений. Общее потребление ресурса увеличится, но не будет превышать $(1+\delta)^{\frac{1}{\kappa}} R = R+\theta$. Длительности работ и максимальное временное смещение для расписания не изменятся. В результате строится расписание с потреблением ресурса не больше $R+\theta$.

Теорема 1. *Для задачи $P|size_j, pmtn, res|L_{\max}$ расписание с потреблением ресурса не более $R+\theta$ можно найти за время, полиномиальное от m, $1/\theta$ и размера входа задачи.*

Задача $P|size_j \in \{1, m\}, pmtn, res|L_{\max}$. Теперь покажем полиномиальную разрешимость случая, когда каждая работа требует либо одну, либо все машины.

Обозначим через $\mathcal{J}_1$ множество одномашинных работ, а через $\mathcal{J}_m$ множество m-машинных работ, $\mathcal{J} = \mathcal{J}_1 \cup \mathcal{J}_m$. Также введем обозначения: $J_{1l} := \{j \in \{l,\ l+1, \ldots, n\} :\ size_j = 1\}$, $J_{ml} := \{j \in \{l,\ l+1, \ldots, n\} : size_j = m\}$ для $l = 1, \ldots, n$, $L_j = \{l = 1, \ldots, n :\ j \in J_{1l} \cup J_{ml}\}$ для $j \in \mathcal{J}$. Введем вещественные переменные $x_{jl} \geq 0$ (длительность работы j в l-м интервале $[d_{l-1}, d_l)$), переменную $L \geq 0$ и построим модель выпуклого программирования:

$$
\begin{gathered}
L \to \min, \\
\tfrac{1}{m}\textstyle\sum_{j\in\mathcal{J}_{11}} p_{jl} \leq d_1 + L - \sum_{j\in\mathcal{J}_{m1}} x_{j1}, \\
\max_{j\in\mathcal{J}_{11}}\{p_{j1}\} \leq d_l + L - \textstyle\sum_{j\in\mathcal{J}_{m1}} x_{j1}, \\
\tfrac{1}{m}\textstyle\sum_{j\in\mathcal{J}_{1l}} p_{jl} \leq d_l - d_{l-1} - \sum_{j\in\mathcal{J}_{ml}} x_{jl},\ l = 2, \ldots, n, \\
\max_{j\in\mathcal{J}_{1l}}\{p_{jl}\} \leq d_l - d_{l-1} - \textstyle\sum_{j\in\mathcal{J}_{ml}} x_{jl},\ l = 2, \ldots, n, \\
\textstyle\sum_{j\in\mathcal{J}_1} \left(\sum_{l\in L_j} p_{jl}\right)^{-1/\kappa} W_j^{1+1/\kappa} + \sum_{j\in\mathcal{J}_m} m \left(\sum_{l\in L_j} p_{jil}\right)^{-1/\kappa} W_j^{1+1/\kappa} \leq R, \\
L \geq 0,\ x_{jl} \geq 0,\ l = 1, \ldots, n,\ j \in \mathcal{J}.
\end{gathered}
$$

Представленную модель можно решить методом эллипсоидов за время, полиномиальное от размера входа задачи. В результате получим максимальное временное смещение и длительности фрагментов x_{jl} для каждой работы j. Затем для каждого интервала строится допустимое расписание, где сначала размещаются m-машинные работы, а потом используется следующий алгоритм для одномашинных работ. В каждом интервале $[d_{l-1} + L + \sum_{j\in\mathcal{J}_{ml}} x_{jl}, d_l + L)$, независимо от остальных ($d_0 = 0$), ставятся в расписание последовательно одномашинные работы, порядок произволен: если некоторая работа не помещается целиком

на текущую машину, то она прерывается и продолжает выполнение на следующей машине.

3. Модель выпуклого программирования и полиномиально разрешимый случай

Предположим что прерывания не допускаются, все работы имеют $size_j = 1,\ j \in J$, и для каждой работы указана машина и позиция, где она выполняется:
n_i — число работ на машине $i \in I$,
π_l^i — работа, выполняемая l-ой по счету на машине i, $l = 1, \ldots, n_i$.

Обозначим через T_j^f переменную, отвечающую за момент завершения работы j, через x_j — переменную, отвечающую за длительность j, $j \in J$, а через L — переменную, соответствующую максимальному временному смещению. Тогда модель выпуклого программирования может быть записана следующим образом:

$$\begin{gathered} L \to \min, \\ L \geq T_j^f - d_j,\ j \in J, \\ T^f_{\pi_l^i} = \textstyle\sum_{k=1}^{l} x_{\pi_k^i}, l = 1, \ldots, n_i,\ i \in I, \\ \textstyle\sum_{j \in J} (W_j)^{1+\frac{1}{\kappa}} (x_j)^{\frac{-1}{\kappa}} \leq R, \\ x_j \geq 0,\ T_j^f \geq 0,\ j \in J. \end{gathered}$$

Представленная задача математического программирования (обозначаемая через M_π) может быть решена за полиномиальное время с помощью метода эллипсоидов [7]. Также для ее решения можно адаптировать более быстрый алгоритм из [2], основанный на использовании условий Каруша – Куна – Таккера.

Как показано в [1], при заданном распределении работ по машинам оптимальное решение соответствует упорядочению работ на машинах по неубыванию директивных сроков и, следовательно, может быть вычислено за полиномиальное время.

Представленный результат справедлив для полностью распараллеливаемых работ, т. е. в случае $size_j = m,\ j \in J$.

4. Задача без прерываний

В общем случае задача без прерываний NP-трудна даже в случае одномашинных работ при $m = 2$ и NP-трудна в сильном смысле при произвольном m. Для доказательства используется сводимость задач Разбиение и 3-Разбиение (веса элементов соответствуют объемам работ,

суммарный объем ресурса равен суммарной длительности работ, число машин соответствует числу подмножеств разбиения).

Для поиска приближенного решения построим алгоритм, основная идея которого заключается в следующем. Пересчитываем объемы работ по правилу $\frac{W_j \cdot size_j}{m}$, уменьшаем общий объем ресурса R в m раз. Решаем задачу на одной машине, что дает нижнюю оценку целевой функции и оценки длительностей работ. Далее составляем расписание с полученными длительностями для распараллеливаемых работ, добавляя работы в расписание на менее загруженные машины по неубыванию их директивных сроков. Детальное описание представлено в схеме $A - m - 1 - m$. Далее покажем, что полученное расписание имеет значение целевой функции $L_{\max}(P) \leq mL^*_{\max}(P1) + (m-1)\max_{j \in J} d_j$, где $L^*_{\max}(P1)$ — оптимум для задачи на одной машине.

Алгоритм $A - m - 1 - m$:

1. Переходим от задачи P на m машинах к задаче $P1$ на одной машине. Объёмы работ W'_j полагаются равными $\frac{W_j \cdot size_j}{m}$, $j \in J$. Объем ресурса уменьшается в m раз, директивные сроки работ не изменяются.
2. Решается задача $P1$ на одной машине, длительности p'_j вычисляются с помощью модели M_π, где работы упорядочены согласно последовательности π по неубыванию директивных сроков.
3. Вычисляются длительности работ для задачи P как $p_j = \frac{p'_j \cdot m}{size_j}$, $j \in J$.
4. Для задачи P работы назначаются в расписание в порядке, полученном в п. 2): работа j назначается в расписание в самый ранний момент времени, когда становится доступным необходимое ей число машин.

Теорема 2. *Если $L_{\max}(P)$ — значение целевой функции задачи P для расписания, построенного алгоритмом $A - m - 1 - m$, $L^*_{\max}(P)$ — оптимальное значение целевой функции для задачи P, то*

$$L_{\max}(P) \leq mL^*_{\max}(P) + (m-1)\max_{j \in J} d_j.$$

Доказательство. Пусть $L_j(P) = c_j(P) - d_j$ — временное смещение для работы $j \in J$ в расписании, построенном для P алгоритмом $A - m - 1 - m$, $L_{\max}(P)$ — значение целевой функции. Здесь и далее $c_j(P)$ ($c_j(P1)$) есть момент окончания работы j в расписании, построенном алгоритмом $A - m - 1 - m$ для задачи P (в оптимальном расписании для $P1$). По построению $L_j(P) = c_j(P) - d_j \leq mc_j(P1) - d_j = m(c_j(P1) - d_j) + (m-1)d_j$. Отметим, что $c_j(P) \leq mc_j(P1)$ для любой работы $j \in J$. Это следует из того, что при переходе к m машинам момент окончания каждой работы будет не больше чем в m раз увеличен, так как длительности увеличиваются не больше чем в m раз.

Тогда для любой работы j имеем $L_j(P) \leq m(c_j(P1) - d_j) + (m-1)d_j$, следовательно, $L_{\max}(P) \leq mL^*_{\max}(P1) + (m-1)\max_{j \in J} d_j$. Из леммы 2 имеем, что $L^*_{\max}(P1) \leq L_{\max}(P1, \sigma_{opt}) \leq L^*_{\max}(P)$, где $L_{\max}(P1, \sigma_{opt})$ —

значение целевой функции для $P1$ при последовательности σ_{opt}, оптимальной для P. □

Лемма 2. *$c'_j(\sigma) \le c_j(\sigma)$, где $c'_j(\sigma)$ — момент окончания j-й работы в задаче $P1$ на одной машине согласно расписанию, соответствующему последовательности завершения работ σ, а $c_j(\sigma)$ — момент окончания j-й работы в задаче P согласно расписанию, соответствующему последовательности завершения работ σ. Длительности работ определяются как в алгоритме $A-m-1-m$.*

Доказательство. Работы добавляются в расписание в одинаковом порядке согласно последовательности σ. В расписании для $P1$ момент окончания j-й работы $c'_j(\sigma) = \sum_{i=1}^{j-1} p'_i + p'_j$.

В расписании для P момент окончания j-й работы

$$c_j(\sigma) \ge \frac{1}{m}\sum_{i=1}^{j-1} p_i \cdot size_i + p_j = \sum_{i=1}^{j-1} p'_i + \frac{p'_j m}{size_j} \ge \sum_{i=1}^{j-1} p'_i + p_j = c'_j(\sigma).$$

□

5. Алгоритмы решения задачи с одномашинными работами

В данном разделе предлагаются и экспериментально исследуются конструктивные и эволюционный алгоритмы для задачи с одномашинными работами ($size_j = 1,\ j \in J$).

Будем использовать следующую кодировку решений $\mathcal{I} =< \{\mathcal{J}_i\}_{i=1}^m, \{R_j\}_{j\in\mathcal{J}} >$, где $\mathcal{J}_i$ — подмножество работ, выполняемых на машине $i = 1, \ldots, m$, а R_j — объем ресурса, потребляемый работой $j \in \mathcal{J}$. При этом должны выполняться условия $\mathcal{J}_i \bigcap \mathcal{J}_k = \varnothing, i \ne k,\ \bigcup_{i=1}^m \mathcal{J}_i = \mathcal{J}$ и $\sum_{j\in\mathcal{J}} R_j \le R$.

Жадные конструктивные алгоритмы. Предлагается два конструктивных алгоритма с общим этапом для распределения работ по машинам и с индивидуальным этапом по распределению ресурса.

Первый этап. Общая эвристика для назначения работ.

Каждой работе $j \in J$ сопоставимо следующее число $rank_j = (W_j)^\kappa - d_j + |\min_{j'\in J}(W_{j'})^\kappa - d_j|$ и будем называть его рангом. Отсортируем работы по неубыванию рангов и $J^* := J$. Для каждой машины i будем хранить текущую сумму рангов, назначенных на нее работ, обозначаемую через S_i. Изначально $S_i = 0$ и при назначении работы j на машину i величина S_i увеличивается на значение $rank_j$. На каждой итерации текущая работа назначается на наименее занятую машину, т. е. $i_{\min} : S_{i_{\min}} = \min_{i\in\{1,\ldots,m\}} S_i$. Трудоемкость шага составляет $O(n\log(n)+nm)$.

Для ранга также рассматривались иные варианты вычисления на основе характеристик работ, но представленная версия показала лучшие результаты.

Второй этап. Распределение ресурса между работами.

Алгоритм 1. Равномерное распределение ресурса $R_j = \frac{R}{n}$ или пропорционально обьемам $R_j = \frac{R \cdot W_j}{\sum_{j' \in J} W_{j'}}$.

Алгоритм 2. Распределение ресурса через решение задачи M_π.

Эволюционный алгоритм. Эволюционный алгоритм имитирует процесс эволюции биологической популяции [8]. Обновление популяции особей (решений) осуществляется посредством операторов селекции, мутации и кроссинговера. Кодировка решения имеет дискретную (распределение работ по машинам) и непрерывную (распределение ресурса между работами) составляющие, а сам алгоритм сочетает в себе принципы генетического алгоритма [12] и дифференциальной эволюции [4].

ЭА: генетический алгоритм и дифференциальная эволюция

Шаг 1. Положить $t := 0$.

Шаг 2. Построить начальную популяцию $\Pi^0 : |\Pi^0| = N$ из особей, сгенерированных случайным образом.

Пока не выполнен критерий остановки, выполнять шаги 3–4.

Шаг 3. Положить $t := t + 1$, $\Pi^{t+1} := \emptyset$.

Шаг 4. Для k от 1 до N выполнять шаги:

Шаг 4.1. Выбрать из Π^t две особи $\mathcal{I}_1, \mathcal{I}_2$ с помощью s-турнирной селекции и одну особь $\mathcal{I}_3$ случайным образом с равномерным распределением.

Шаг 4.2. С вероятностью P_γ применить оператор кроссинговера к $\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3$ и построить особь $\mathcal{I}'$.

Шаг 4.3. Иначе с вероятностью $(1 - P_\gamma)$ применить оператор мутации к $\mathcal{I}_3$ и построить особь $\mathcal{I}'$.

Шаг 4.4. Положить $\Pi^{t+1} := \Pi^{t+1} \cup \mathcal{I}'$.

Шаг 5. Результатом работы алгоритма является лучшая из найденных особей.

В операторе s-**турнирной селекции** из популяции с равномерным распределением извлекается s особей и из них выбирается лучшая.

Оператор **мутации** состоит из двух этапов: перемещение работ и перемещение ресурса. При перемещении работ для каждой работы $j \in \mathcal{J}$ с вероятностью P_ξ случайным образом выбирается новая машина с равномерным распределением.

При перемещении ресурса используется параметр $0 < \delta \leq 1$ ($\delta := 0,7\frac{k}{N}$ на основе предварительных вычислительных экспериментов). Для каждой работы $j \in \mathcal{J}$ с вероятностью P_η проводится следующая процедура: выбирается случайное число Δ из $(0, \delta]$ и случайное целое число r из $\{1, \ldots, n\}$ и полагается $R_j = (1 - \Delta)R_j$, $R_r = R_r + \Delta R_j$. Параметр δ — верхняя граница доли перемещаемого ресурса, меняется в зависимости от итерации цикла. Варьирование параметра δ помогает алгоритму настраивать глубину поиска.

В операторе **кроссинговера** также происходит скрещивание по двум компонентам решений. При скрещивании по компоненте с работами расположение работы на машине определяется следующим образом: если в родительских особях работа расположена на одной и той же машине, то работа остается на этой машине, иначе случайным образом с равной вероятностью выбирается одна из родительских машин.

При скрещивании компонент с ресурсом оператор работает по правилу. Ресурс, назначаемый работе $j \in J$, определяется формулой

$R_j = \lambda' R_j^3 + (1-\lambda')(\lambda R_j^1 + (1-\lambda)R_j^2),\ 0 < \lambda' < 1, 0 < \lambda < 1.$

Параметр λ отвечает за баланс ресурса, взятого от основных двух выбранных родительских особей. Параметр λ' и случайным образом выбранная особь $\mathcal{I}^3 \in \Pi^t$ помогают алгоритму более эффективно искать распределение ресурса, добавляя в кроссинговер больше информации о распределении ресурса в других особях из популяции. Однако, как показали результаты предварительного вычислительного эксперимента, при значениях параметра λ' выше $0,15$ качество результатов работы алгоритма начинает падать, возможно, в силу существенной случайности при расчете объема назначаемого ресурса.

Также был апробирован ЭА, где в кодировке решений фигурирует только распределение работ по машинам, а распределение ресурса осуществляется посредством решения задачи M_π с помощью пакетов прикладных программ. Такая версия алгоритма показала менее перспективные результаты в вычислительном эксперименте.

Вычислительный эксперимент. Тестовые примеры для вычислительного эксперимента взяты из открытой библиотеки примеров OR-Library [3] и адаптированы к исследуемой задаче с $n = 50,\ 100$ при $R = \sum_{j \in \mathcal{J}} W_j^{\frac{\kappa-1}{\kappa}}$. При настройке параметров ЭА рассматривались варианты: размер популяции N из множества $\{50,\ 100,\ 200,\ 300,\ 350\}$; вероятность перемещения работы P_ξ, вероятность мутации для ресурса работы P_η, вероятность применения оператора кроссинговера P_γ из множества $\{\frac{1}{10},\ \frac{1}{9},\ \frac{1}{8}, \ldots, \frac{1}{4}\}$.

В табл. 1 представлены результаты вычислительного эксперимента для $n = 50$ и $n = 100$, где сравниваются жадный алгоритм 1 и ЭА. На основе предварительного вычислительного эксперимента для ЭА выбраны следующие значения параметров для задач библиотеки с $n = 50$: $N = 100$, $P_\xi = \frac{1}{6}$, $P_\eta = \frac{1}{4}$, $P_\gamma = \frac{1}{7}$, число итераций до остановки ЭА $T = 1000$. Заметим, что ЭА показывает статически значимое преимущество над жадным алгоритмом для всех рассматриваемых случаев с различным числом машин m и значением параметра κ для учета ресурса (полученные решения сравниваются с помощью критерия Вилкоксона). Для $m = 1, \kappa = 0,8$ ЭА оказался в среднем на 97 % лучше, чем жадный алгоритм, а для $m = 2$ и 5 показал в среднем на 25 % лучше

результаты. По мере уменьшения значения κ относительное отклонение результата работы жадного алгоритма 1 от ЭА уменьшается.

Таблица 1.

Относительное отклонение в % результатов жадного алгоритма 1 от результатов ЭА на сериях примеров с $n = 50$ и $n = 100$

		n = 50			n = 100		
m	κ	min	max	avg	min	max	avg
1	0,3	0,50	242,91	9,61	0,15	73,90	5,37
	0,5	3,68	72,44	12,85	2,56	183,59	11,73
	0,8	10,30	488,76	97,23	6,18	260,32	90,75
2	0,3	0,37	9,21	2,33	0,15	489,46	16,71
	0,5	1,60	124,22	25,00	0,99	1126,15	47,62
	0,8	6,44	125,67	23,10	2,71	49,52	16,44
5	0,3	0,40	427,02	25,98	0,15	3291,31	62,36
	0,5	1,20	534,76	31,95	0,70	3229,91	72,27
	0,8	2,58	660,59	31,10	0,72	2514,94	68,37

Теперь рассмотрим результаты эксперимента для задач библиотеки с $n = 100$. На основе предварительного эксперимента выбраны значения параметров ЭА: при $m = 1$ размер популяции $N = 100$, максимальное число итераций $T = 1000$, при $m > 1$ размер популяции $N = 350$, максимальное число итераций $T = 800$; вероятности $P_\xi = \frac{1}{6}$, $P_\eta = \frac{1}{4}$, $P_\gamma = \frac{1}{7}$. ЭА показал статистически значимое преимущество над жадным алгоритмом по всем сериям. Также заметим, что на отдельных задачах ЭА находил существенно лучшее решение, в несколько раз превышающее результат жадного алгоритма. Для $m = 1, \kappa = 0,8$ ЭА показал результаты в среднем на 90,75

Аналогичные результаты наблюдаются при иных рассмотренных значениях параметров, ухудшаясь с ростом вероятности мутации. ЭА также имеет статистически значимое преимущество над жадным алгоритмом, демонстрируя результаты не более чем на 10

В табл. 2 представлено сравнение результатов жадного алгоритма 2 и ЭА на сериях с числом работ $n = 50$ и $n = 100$. В качестве решателя задачи выпуклого программирования в жадном алгоритме использовался солвер OSQP библиотеки **cvxpy** языка программирования Python. В случае $m = 1$ жадный алгоритм находил точное решение задачи. ЭА ошибался в среднем не более чем на 3,23 % при $n = 50$ (8,88 % при $n = 100$) в случае сравнения с точным решением. При $m > 1$ ЭА показал статистически значимое превосходство над жадным алгоритмом при $n = 50$. В таблице отсутствует относительное отклонение для $m > 1$ при $n = 100$, потому что решение задачи выпуклого программирования требовало значительных вычислительных ресурсов и более 7 ГБ

оперативной памяти. Этот факт также подчеркивает обоснованность использования ЭА при решении исследуемой задачи.

Таблица 2.

Относительное отклонение в % жадного алгоритма распределением ресурса МВП от ГА с дифференциальной эволюцией на серии из 60 задач. $n = 50$

		n = 50			n = 100		
m	κ	min	max	avg	min	max	avg
1	0,3	-0,01	-8,47	-0,38	-0,018	-11,78	-0,86
	0,5	-0,05	-3,64	-0,31	-0,24	-18,73	-1,55
	0,8	-0,14	-10,89	-3,23	-0,62	-23,47	-8,88
2	0,3	0,31	7,61	1,07	–	–	–
	0,5	0,02	38,44	2,01	–	–	–
	0,8	4,57	29,86	5,88	–	–	–
5	0,3	0,01	40,84	4,35	–	–	–
	0,5	0,04	97,71	6,30	–	–	–
	0,8	0,02	576,13	16,95	–	–	–

6. Заключение

Проведено исследование задачи минимизации максимального временного смещения многомашинных работ в случае наличия ресурса, влияние которого выражается выпуклой функцией. Доказано, что в случае задачи с прерываниями «почти точное» решение может быть найдено за полиномиальное время при ограниченном константой числе машин или когда работы задействуют одну или все машины. Для задачи без прерываний разработан алгоритм с гарантированной оценкой точности. Выделены полиномиально разрешимые случаи. Для одномашинных работ предложены и экспериментально исследованы эвристические алгоритмы. Для дальнейших исследований представляет интерес разработка метаэвристик для задачи с многомашинными работами.

Список источников

1. Захарова Ю. В., Евтина А. О. Конструктивные алгоритмы для задачи составления расписаний на двух процессорах с критерием максимального временного смещения при учете распараллеливания и расхода энергии // Прикладная дискретная математика. 2025. № 67. С. 118–128.
2. Speed scaling for maximum lateness / E. Bampis, D. Letsios, I. Milis, G. Zois // International Computing and Combinatorics Conference. Berlin, Heidelberg : Springer, 2012. P. 25–36. https://doi.org/10.1007/978-3-642-32241-9_3
3. Beasley J. E. OR-library: Weighted tardiness.
URL: https://people.brunel.ac.uk/ mastjjb/jeb/orlib/wtinfo.html

4. Caponio A., Neri F., Tirronen V. Super-fit control adaptation in memetic differential evolution frameworks // Soft Computing. 2009. Vol. 13. P. 811–831.
5. Garey M. R., Johnson D. S. Computers and Intractability: A Guide to the Theory of NP-Completeness. San Francisco, Calif. : W. H. Freeman and Co., 1979. 338 p.
6. Gerards M. E. T., Hurink J. L., Holzenspies P. K. F. A survey of offline algorithms for energy minimization under deadline constraints // J. Scheduling. 2016. Vol. 19. P. 3–19. https://doi.org/10.1007/s10951-015-0463-8
7. Grötschel M., Lovász L., Schrijver A. Geometric Algorithms and Combinatorial Optimizations. 2nd corrected ed. Berlin : Springer-Verlag, 1993. 362 p.
8. Holland J. H. Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. MIT press, 1992. 211 p.
9. Kononov A., Kovalenko Y. On speed scaling scheduling of parallel jobs with preemption // DOOR-2016, LNCS. 2016. Vol. 9869. P. 309–321. https://doi.org/10.1007/978-3-319-44914-2_25
10. Kononov A., Kovalenko Y. Approximate schedules for non-migratory parallel jobs in speed-scaled multiprocessor systems // Siberian Electronic Mathematical Reports. 2019. Vol. 16. P. 249–257. https://doi.org/10.33048/semi.2019.16.016
11. Kononov A., Zakharova Y. Speed scaling scheduling of multiprocessor jobs with energy constraint and makespan criterion // J. Glob. Optim. 2022. Vol. 83. P. 539–564. https://doi.org/10.1007/s10898-021-01115-x
12. Genetic algorithms for planning and scheduling engineer-to-order production: A systematic review / A. Neumann, A. Hajji, M. Rekik, R. Pellerin // International Journal of Production Research. 2024. Vol. 62, N 8. P. 2888–2917.
13. Shabtay D., Kaspi M. Parallel machine scheduling with a convex resource consumption function // European Journal of Operational Research. 2006. Vol. 173, N 1. P. 92–107. https://doi.org/10.1016/j.ejor.2004.12.008
14. Shabtay D., Kaspi M. Minimizing the makespan in open–shop scheduling problems with a convex resource consumption function // Naval Research Logistics. 2006. Vol. 53, N 3. P. 204–216. https://doi.org/10.1002/nav.20133
15. Zakharova Y. V. Approximation algorithms for Open Shop variations subject to energy consumption // Proceedings of the Steklov Institute of Mathematics. 2024. Vol. 327. P. S286–S301. https://doi.org/10.1134/S0081543824070216

References

1. Zakharova Y.V., Evtina A.O. Constructive algorithms for the scheduling problem on two processors with the maximum time offset criterion taking into account parallelization and energy consumption. *Prikl. Diskr. Mat.*, 2025, no. 67, pp. 118–128. https://doi.org/10.17223/20710410/67/7 (in Russian)
2. Bampis E., Letsios D., Milis I., Zois G. Speed scaling for maximum lateness. *International Computing and Combinatorics Conference.* Berlin, Heidelberg, Springer, 2012, pp. 25–36. https://doi.org/10.1007/978-3-642-32241-9_3
3. Beasley J.E. OR-library: weighted tardiness. Available at: https://people.brunel.ac.uk/ mastjjb/jeb/orlib/wtinfo.html
4. Caponio A., Neri F., Tirronen V. Super-fit control adaptation in memetic differential evolution frameworks. *Soft Computing*, 2009, vol. 13, pp. 811–831.
5. Garey M.R., Johnson D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* San Francisco, Calif., W. H. Freeman and Co., 1979, 338 p.

6. Gerards M.E.T., Hurink J.L., Holzenspies P.K.F. A survey of offline algorithms for energy minimization under deadline constraints. *J. Scheduling*, 2016, vol. 19, pp. 3–19. https://doi.org/10.1007/s10951-015-0463-8
7. Grötschel M., Lovász L., Schrijver A. *Geometric Algorithms and Combinatorial Optimizations, 2nd correct. ed.* Berlin, Springer-Verlag, 1993, 362 p.
8. Holland J. H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence.* MIT press, 1992. 211 p.
9. Kononov A., Kovalenko Y. On speed scaling scheduling of parallel jobs with preemption. *DOOR-2016*, LNCS, 2016, vol. 9869, pp. 309–321. https://doi.org/10.1007/978-3-319-44914-2_25
10. Kononov A., Kovalenko Y. Approximate schedules for non-migratory parallel jobs in speed-scaled multiprocessor systems. *Siberian Electronic Mathematical Reports*, 2019, vol. 16, pp. 249-257. https://doi.org/10.33048/semi.2019.16.016
11. Kononov A., Zakharova Y. Speed scaling scheduling of multiprocessor jobs with energy constraint and makespan criterion. *J. Glob. Optim.,* 2022, vol. 83, pp. 539–564. https://doi.org/10.1007/s10898-021-01115-x
12. Neumann A., Hajji A., Rekik M., Pellerin R. Genetic algorithms for planning and scheduling engineer-to-order production: A systematic review. *International Journal of Production Research,* 2024, vol. 62, no. 8, pp. 2888–2917.
13. Shabtay D., Kaspi M. Parallel machine scheduling with a convex resource consumption function. *European Journal of Operational Research,* 2006, vol. 173, no. 1, pp. 92–107. https://doi.org/10.1016/j.ejor.2004.12.008
14. Shabtay D., Kaspi M. Minimizing the makespan in open–shop scheduling problems with a convex resource consumption function. *Naval Research Logistics*, 2006, vol. 53, no. 3, pp. 204–216. https://doi.org/10.1002/nav.20133
15. Zakharova Y.V. Approximation algorithms for Open Shop variations subject to energy consumption. *Proceedings of the Steklov Institute of Mathematics*, 2024, vol. 327, pp. S286–S301. https://doi.org/10.1134/S0081543824070216

Об авторах

Захарова Юлия Викторовна, канд. физ.-мат. наук, Омский филиал Института математики им. С. Л. Соболева СО РАН, Омск, 644099, Российская Федерация, yzakharova@ofim.oscsbras.ru, https://orcid.org/0000-0003-4791-7011

Городецкий Дмитрий Дмитриевич, Омский государственный университет им. Ф. М. Достоевского, Омск, 644077, Российская Федерация, gorodeckiydimchik@gmail.com

About the authors

Yulia V. Zakharova, Cand. Sci. (Phys.-Math.), Omsk Branch of the Sobolev Institute of Mathematics SB RAS, Omsk, 644099, Russian Federation, yzakharova@ofim.oscsbras.ru, https://orcid.org/0000-0003-4791-7011

Dmitry D. Gorodecky, Dostoevsky Omsk State University, Omsk, 644077, Russian Federation, gorodeckiydimchik@gmail.com

Захаров Алексей Олегович,
канд. физ.-мат. наук, Омский филиал Института математики им. С. Л. Соболева СО РАН, Омск, 644099, Российская Федерация,
azakharov@ofim.oscsbras.ru,
https://orcid.org/0000-0003-2469-023X

Aleksey O. Zakharov, Cand. Sci. (Phys.-Math.), Omsk Branch of the Sobolev Institute of Mathematics SB RAS, Omsk, 644099, Russian Federation,
azakharov@ofim.oscsbras.ru,
https://orcid.org/0000-0003-2469-023X

Поступила в редакцию / Received 15.10.2025
Поступила после рецензирования / Revised 22.12.2025
Принята к публикации / Accepted 24.12.2025