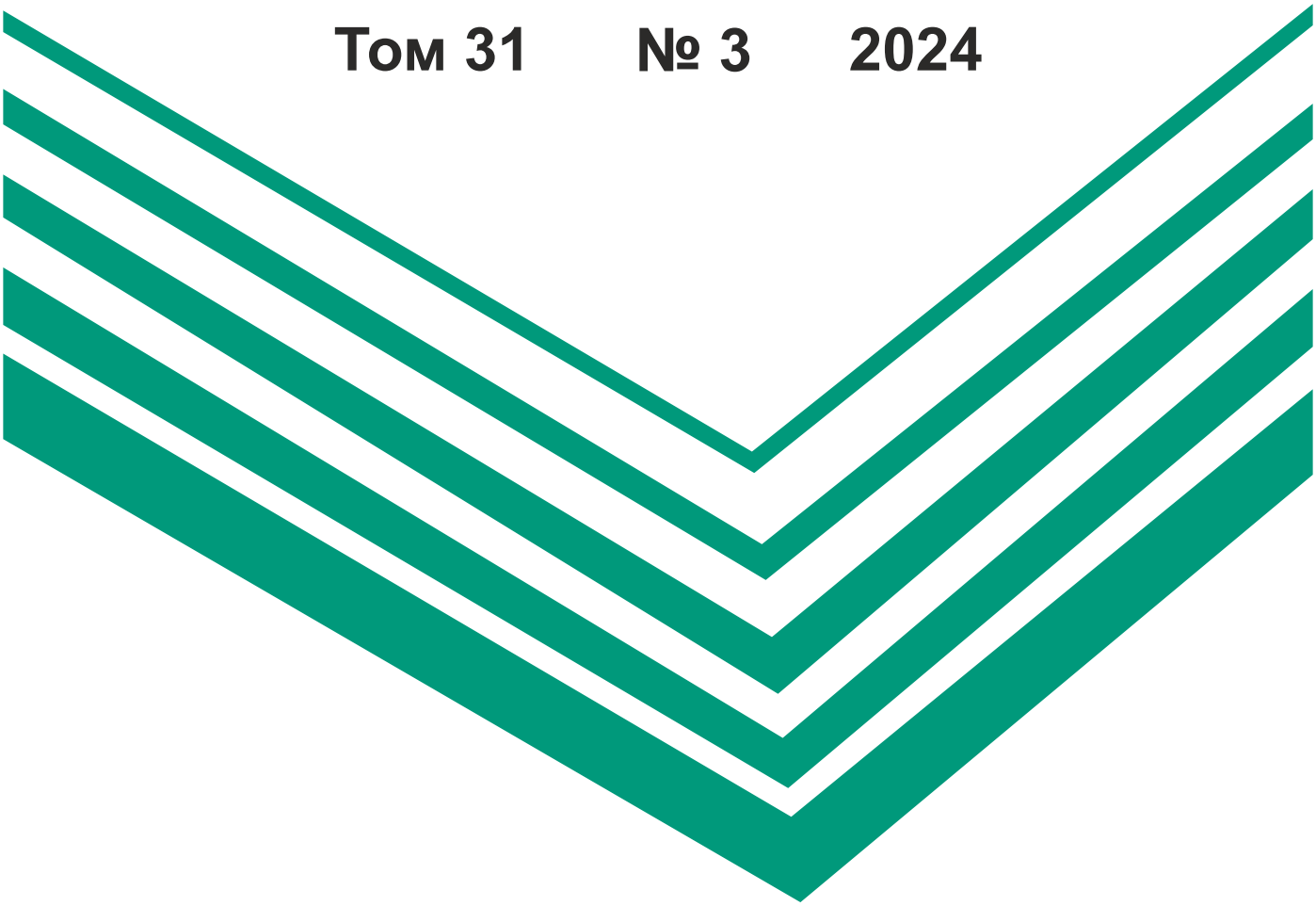
ISSN 2949-5598

ДИСКРЕТНЫЙ АНАЛИЗ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ

Том 31 № 3 2024



Новосибирск
Издательство Института математики

ДИСКРЕТНЫЙ АНАЛИЗ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ
Июль–сентябрь 2024. Т. 31, № 3. С. 105–122

УДК 519.8+518.25 DOI: 10.33048/daio.2024.31.750

ЗАДАЧА ОДНОГО СТАНКА С РАВНЫМИ ДЛИТЕЛЬНОСТЯМИ РАБОТ И ВОЗМОЖНОСТЬЮ ПРЕРЫВАНИЙ

К. А. Ляшкова[a], *В. В. Сервах*[b]

Омский филиал Института математики им. С. Л. Соболева,
ул. Певцова, 13, 644099 Омск, Россия
E-mail: [a] ksech@bk.ru, [b] svv_usa@rambler.ru

Аннотация. Рассматривается задача минимизации среднего взвешенного времени для выполнения работ одинаковой длительности на одном станке при заданных временах поступления работ и возможности их прерывания. В настоящее время вычислительная сложность этой задачи неизвестна. В работе предложен алгоритм предобработки входных данных, что позволяет свести задачу к более узкому и регулярному классу примеров. Обоснованы свойства оптимальных решений, на основе которых разработан алгоритм построения конечного подмножества решений, содержащего оптимальное расписание. Описан подход к проведению параметрического анализа расписаний из этого подмножества, который позволяет сформировать подкласс расписаний, оптимальных при некоторых значениях весов. Выделен полиномиально разрешимый случай задачи. Табл. 1, ил. 10, библиогр. 16.

Ключевые слова: теория расписаний, один станок, прерывание.

1. Постановка задачи

На единственном станке необходимо выполнить n работ. Заданы моменты поступления работ в систему r_i, длительности их выполнения p_i и веса ω_i, $i = 1, 2, \ldots, n$. Станок в каждый момент времени может выполнять только одну работу. Требуется найти расписание выполнения работ, при котором взвешенная сумма моментов их завершения будет наименьшей. Время завершения работы i будем обозначать через C_i. Тогда целевая функция выглядит следующим образом: $\sum\limits_{i=1}^{n} \omega_i C_i$. В общепринятых обозначениях, например [1], эта задача записывается как $1|r_i|\sum \omega_i C_i$.

Задача NP-трудна в сильном смысле даже в случае единичных весов [2]. В [3] доказана полиномиальная разрешимость задачи $1|r_i, p_i =$

$p|\sum \omega_i C_i$ с одинаковыми длительностями работ. Если допустить прерывания, то задача $1|r_i, pmtn|\sum \omega_i C_i$ остаётся сильно NP-трудной [4]. Для единичных весов задача $1|r_i, pmtn|\sum C_i$ полиномиально разрешима [5]. Вопрос о вычислительной сложности задачи с прерываниями и равными длительностями работ $1|r_i, p_i = p, pmtn|\sum \omega_i C_i$ остаётся открытым.

Подробное исследование задач теории расписаний с возможностью прерывания работ проведено в [6]. В [7, 8] описаны некоторые важные свойства и, в частности, тот факт, что для поиска оптимума достаточно рассматривать расписания, в которых работа прерывается только в момент поступления работы с бóльшим весом. В [9] предложены релаксационные модели и градиентные алгоритмы с оценками точности получаемого решения для задачи $1|r_i|\sum \omega_i C_i$. Отметим также работы [10, 11], посвящённые исследованию модели линейного целочисленного программирования указанной задачи. В [12, 13] описан параметрический подход к исследованию комбинаторной структуры возможных оптимальных решений $1|r_i, p_i = p, pmtn|\sum \omega_i C_i$. Некоторые свойства оптимальных решений, полученные на основе параметрического анализа, изложены в [14]. В [15, 16] рассматривается задача со штрафами за нарушения директивных сроков выполнения работ и возможностью сверхурочных работ на станке.

В разд. 2 предложен алгоритм предобработки входных данных, который позволяет свести задачу к решению примеров более простой и регулярной структуры. В разд. 3 обоснованы некоторые свойства, на основе которых может быть построено конечное подмножество решений, включающее оптимальное расписание. Изложен алгоритм его формирования. В разд. 4 предлагается подход к параметрическому анализу расписаний этого подмножества. В результате такого анализа выделяется подкласс расписаний, которые являются оптимальными при некоторых значениях весов. В разд. 5 описан полиномиально разрешимый случай задачи.

2. Предобработка входных данных

В данном разделе задачу с прерываниями $1|r_i, p_i = p, pmtn|\sum \omega_i C_i$ сведём к совокупности подзадач более простой и регулярной структуры. Через S_i обозначим время начала выполнения работы $i = 1, 2, \ldots, n$. Прежде всего отметим свойство, связанное с порядком выполнения работ.

Утверждение 1. *Если $\omega_i < \omega_j$ и $r_i \geqslant r_j$, то для поиска оптимального решения достаточно рассматривать расписания, в которых работа j целиком выполняется раньше работы i, т. е. $C_j \leqslant S_i$.*

Доказательство. Рассмотрим произвольное расписание и выделим в нём совокупность всех интервалов, в течение которых выполняются

работы i и j. Суммарная длина этих интервалов $2p$. Составим новое расписание, выполняя в этих временны́х интервалах сначала работу j, а потом i. Остальные работы такая перестановка не затронет, и целевая функция при этом не увеличится. Тем самым получим расписание, для которого утверждение справедливо. Утверждение 1 доказано.

Значит, при $\omega_i < \omega_j$ и $r_i \geqslant r_j$ порядок выполнения работ i и j известен. Проблема выбора возникает при условии $r_i < r_j$ и $\omega_i < \omega_j$, когда работа с бóльшим весом поступает позднее. Такие работы назовём *конкурентными*. Заметим, что оба неравенства строгие, так как если выполняется хотя бы одно равенство, то работы не будут конкурентными. Если $r_i = r_j$ и $\omega_i = \omega_j$, то работы идентичны. В такой ситуации более приоритетной будет считаться работа с бóльшим номером.

Утверждение 2. *Для поиска оптимального решения достаточно рассматривать расписания, в которых интервал $[S_j, C_j]$ не содержит фрагментов работы с весом, меньшим ω_j.*

Доказательство. Если есть работа j, которая прерывает работу с меньшим весом, то работа j завершается до того, как работа с меньшим весом возобновит своё выполнение. Это означает, что работа i с меньшим весом либо выполняется целиком до начала работы j с бóльшим весом, либо целиком после, либо окаймляет её. Этот факт подробно исследован в [6]. Утверждение 2 доказано.

Далее опишем некоторые предварительные процедуры обработки данных, позволяющие свести задачу $1|r_i, p_i = p, pmtn|\sum \omega_i C_i$ к решению серии задач более простой структуры.

Упорядочим работы по неубыванию весов $\omega_1 \leqslant \omega_2 \leqslant \ldots \leqslant \omega_n$. Рассмотрим две неконкурентные работы i и j, для которых $\omega_i \leqslant \omega_j$ и $r_i \geqslant r_j$. По утверждению 1 работа j целиком выполняется раньше работы i, значит, $C_j \leqslant S_i$. Если интервалы $[r_j, r_j + p]$ и $[r_i, r_i + p]$ пересекаются, т. е. $r_j \leqslant r_i < r_j + p$, то раньше момента $r_j + p$ работа i начаться не может, поэтому значение r_i можно увеличить до $r_j + p$. Если работы идентичны, то приоритет отдаём работе с бóльшим номером. Сделав это для каждой пары неконкурентных работ с пересекающимися интервалами $[r_j, r_j + p]$ и $[r_i, r_i + p]$ (возможно неоднократно), придём к ситуации, когда все r_i будут различны. Напомним, что если $r_i = r_j$, то работы i и j неконкурентны, и к ним применима эта процедура.

Результаты предобработки отображены на рис. 1 и 2. Под каждую работу отводим временну́ю ось и эти оси располагаем одну под другой, начиная с работы с наибольшим весом и далее по убыванию весов. Моменты поступления работ выделены жирной чертой. На рис. 1 изображён

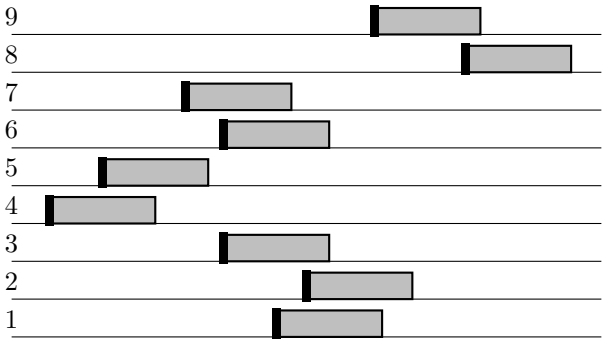


Рис. 1. Входные данные

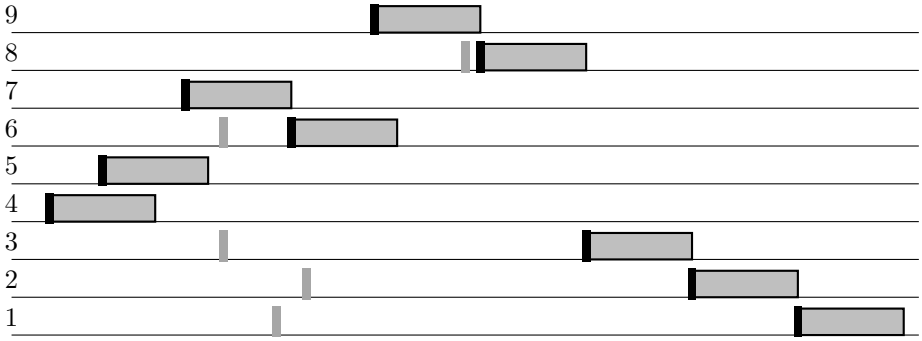


Рис. 2. Входные данные после предобработки

пример входных данных. Здесь, как и ранее, $\omega_1 \leqslant \omega_2 \leqslant \ldots \leqslant \omega_9$. Работы отображены интервалами $[r_i, r_i + p]$. На рис. 2 представлены входные данные после предобработки.

Далее рассмотрим процедуру декомпозиции задачи. Напомним, что работа с номером 1 имеет самый маленький вес. Выделим подмножество работ, которые поступают не позже r_1. Пусть их k штук. По утверждению 1 все эти работы завершаются до начала работы 1, но тогда она не может начинаться раньше момента pk, и можно положить $r_1 = pk$. Далее повторим эту процедуру, пока ранний срок начала выполнения работы 1 можно будет сдвигать.

Если работа 1 сдвинется в самый конец, она завершится в момент pn. Тем самым она не влияет на выполнение других работ, и получаем задачу меньшей размерности. В приведённом выше примере (рис. 1 и 2) задача с 8 работами в итоге сводится к задаче с пятью работами $\{4, 5, 6, 7, 8\}$.

Пусть r_1 увеличилось до pk, $k < n$. Тогда выполнение работы 1 однозначно начинается в этот момент, так как альтернативных работ нет.

Если такие были бы, то работа 1 сдвинулась бы ещё дальше, т. е. для всех остальных работ $r_i > pk$. В этом случае задачу можно разбить на две подзадачи: в первую войдут k работ с моментами $r_i \leqslant p(k-1)$, во вторую — все остальные, включая работу 1, причём работа 1 поступает раньше всех. Кроме того, отметим, что если во второй подзадаче все $r_i \geqslant pk + p$, то работа 1 полностью выполняется в интервале $[pk, pk + p]$, и можно рассматривать задачу без неё, т. е. задача вновь разбивается. Декомпозиции не будет, если отрезки $[r_1, r_1 + p]$ и $[r_i, r_i + p]$ пересекаются.

Отметим, что в первой подзадаче, которая решается на интервале $[0, kp]$, вновь можно применить этот подход со сдвигом работы с меньшим весом, если она не поступает первой. Таким образом, исходная задача разбивается на задачи меньшей размерности, в каждой из которых первой поступает работа с наименьшим весом, а до её завершения поступает следующая работа.

Развивая описанный подход далее, можно утверждать, что j-я по порядку поступления в систему работа должна поступить не позднее момента $p(j-1) - 1$. Иначе все предшествующие $j - 1$ работ успевают завершить своё выполнение и к задаче снова можно применить декомпозицию. Задачу, которую нельзя упростить указанными процедурами, будем называть *приведённой*. В приведённой задаче

- работа с наименьшим весом поступает первой в нулевой момент времени;
- все времена поступлений r_i, $i = 1, 2, \ldots, n$, различны;
- j-я по порядку поступления работа поступает не позднее момента $p(j-1) - 1$, $j = 2, 3, \ldots, n$;
- достаточно рассматривать расписания длины pn, в которых простои станка отсутствуют.

Далее будем работать только с приведёнными расписаниями.

3. Построение конечного множества расписаний

В этом разделе опишем процедуру выделения конечного подмножества расписаний, которое содержит оптимальное.

Утверждение 3. *Для построения оптимального решения достаточно рассмотреть расписания, в которых переключение станка на выполнение другой работы происходит только в момент r_i или в момент C_j окончания некоторой работы.*

Доказательство. В момент окончания некоторой работы станок освобождается и, естественно, переходит в состояние простоя или выполнения другой доступной работы. Предположим, что переключение произошло в момент, не совпадающий с некоторым r_i, и станок переключился с выполнения работы j на выполнение работы k. В этом случае

в соответствии с [6] работа j окаймляет работу k. Тогда найдётся $\varepsilon > 0$ такое, что некоторый фрагмент работы j длины ε, предшествующий моменту прерывания, можно поменять с фрагментом работы k в интервале $[C_k - \varepsilon, C_k]$. В этом случае время завершения работы k, а значит, и значение целевой функции уменьшатся. Параметр ε может быть выбран как минимальное значение трёх величин: длины фрагмента работы j, предшествующего прерыванию, длины заключительного фрагмента работы k и резерва сдвига работы k, равного $S_k - r_k$. Утверждение 3 доказано.

Отсюда вытекает, что в приведённой задаче существует оптимальное расписание, в котором все моменты прерываний работ принадлежат множеству $\{r_2, r_3, \ldots, r_n\}$, а суммарное число переключений станка не превосходит $2n-2$. Далее будем иметь дело только с такими расписаниями. Рассмотрим возможные альтернативы выбора работ в моменты переключения станка.

Утверждение 4. *Для построения оптимального решения достаточно рассмотреть расписания, в которых прерывание текущей работы в момент r_j происходит только в том случае, когда поступившая работа имеет наибольший вес среди всех незавершённых и доступных для выполнения работ. При этом в момент r_j начинает выполнение работа j.*

Доказательство. Первая часть утверждения очевидна, поскольку если вес работы j меньше веса некоторой ранее поступившей работы, то по утверждению 1 эта работа должна быть завершена раньше j. Доказательство второй части утверждения 4 аналогично обоснованию утверждения 3. Если текущая работа прервалась работой k, для которой $r_k < r_j$, то фрагменты работы k и текущей работы могут быть переставлены, как ранее. При этом целевая функция уменьшится.

Таким образом, в момент r_j имеет место альтернатива выбора между текущей работой i и поступившей работой j. При этом если выбор происходит в пользу работы j, то она целиком завершится до возобновления работы i. Если выбирается работа i, то до её завершения работа j не начнёт выполняться. Утверждение 4 доказано.

Опишем, какие есть варианты продолжения расписания в момент C_j окончания некоторой работы.

1. Все работы завершили своё выполнение. Расписание построено.

2. Не все работы выполнены, но в момент времени C_j незавершённых и готовых к выполнению работ нет. Тогда станок будет простаивать до поступления следующей работы. Такая ситуация не возникает, если мы решаем приведённую задачу.

3. Есть доступные работы, незавершённых работ нет. Выполнение начнёт работа с наибольшим весом.

4. Есть незавершённые работы, при этом все доступные уже начаты. Тогда продолжит выполнение последняя прерванная работа.

5. Есть и прерванные, и доступные работы. Если i — последняя из прерванных работ, j — доступная работа с наибольшим весом и при этом $i > j$, то продолжаем выполнять работу i.

6. Наконец, есть и прерванные, и доступные работы, i — последняя из прерванных работ, j — доступная работа с наибольшим весом и при этом $i < j$. Тогда возникает альтернатива выбора между i и j, т. е. либо продолжит выполнение последняя из прерванных работ, либо начнёт выполнение доступная работа с максимальным весом.

Опишем алгоритм формирования всех расписаний, удовлетворяющих описанным выше свойствам. Работы упорядочены в лексикографическом возрастающем порядке векторов (ω_1, r_i). Алгоритм заключается в последовательном просмотре моментов r_i и C_i в порядке их возрастания, и ветвлении вариантов в случае альтернативы выбора работ. Напомним, что в момент r_i возможна альтернатива между текущей работой и поступившей работой i, а в C_i — между последней прерванной работой и самой тяжёлой из доступных, но не начавших выполнения работ.

Первоначально формируется стек T из работ, находящихся в процессе выполнения и множество работ D, доступных для выполнения. Стек T организуем по стандартному правилу «последний зашёл — первый вышел». Первоначально полагаем $T = (1)$ и $D = \varnothing$. Получаем некоторое частичное расписание.

Очередной шаг. Рассматриваем любое из построенных частичных расписаний. Пусть в этом расписании часть работ уже завершена, часть прервана, и выполняется работа k, которая стоит последней в стеке T. Выполняем эту работу до момента C_k или ближайшего r_m. Если $C_k \leqslant r_m$ или все работы уже поступили в систему, то работу k из T исключаем. При $r_m \leqslant C_k$, в D добавляем работу m.

Далее в T берём последний элемент (пусть это будет номер i), а в D — элемент с наибольшим весом (пусть это j). Если $i > j$, то в соответствии со случаем 5 продолжаем выполнять работу i, иначе ветвим. В одном варианте выполняем работу i. Тогда T и D не меняем. Во втором выполняем работу j. Её надо удалить из D и поместить в стек T. Получили два новых варианта частичных расписаний. Переходим на очередной шаг.

Построение любого частичного расписания завершается, когда D и T становятся пустыми.

Число построенных расписаний обозначим через Q. Так как выбор осуществляется только в моменты r_j, кроме минимального, и в моменты C_j, кроме двух последних, может быть сгенерировано не более 2^{2n-3} различных расписаний, но это грубая оценка. На самом деле значение Q меньше, поскольку ветвление происходит только тогда, когда и стек,

и очередь одновременно не пустые. Максимум Q достигается на примерах, когда все пары работ конкурентные и работы поступают в систему равномерно через промежуток времени $\frac{p}{2}$. Анализ результатов работы программы показывает, что в этом случае $Q = O(e^n)$.

Далее проведём анализ расписаний построенного множества с учётом весов работ.

4. Параметрический анализ

Зафиксируем значения p и r_i, $i = 1, 2, \ldots, n$, и описанным выше алгоритмом построим конечное множество расписаний. При формировании этого множества не учитываются значения весовых коэффициентов ω_i, $i = 1, 2, \ldots, n$. Их будем рассматривать как параметры. Без ограничения общности можно считать, что $0 < \omega_1 \leqslant \omega_2 \leqslant \ldots \leqslant \omega_n$. Более того, веса можно ограничить, положив $\omega_n = 1$.

Пусть C_i^q — момент завершения работы i в расписании $q = 1, 2, \ldots, Q$. Чтобы расписание q_0 было оптимальным, необходимо и достаточно, чтобы была совместна следующая система неравенств:

$$\sum_{i=1}^{n} C_i^{q_0}\omega_i \leqslant \sum_{i=1}^{n} C_i^{q}\omega_i, \quad q = 1, 2, \ldots, Q, \ q \neq q_0,$$
$$0 < \omega_1 \leqslant \omega_2 \leqslant \ldots \leqslant \omega_n = 1.$$

Проверяя совместность системы для каждого из Q сформированных расписаний, выделим подмножество расписаний, оптимальных при некоторых значениях весов. Обозначим его через OPT. Отметим некоторую особенность формирования этого подмножества. В единичном $(n-1)$-мерном кубе рассмотрим область $0 < \omega_1 \leqslant \omega_2 \leqslant \ldots \leqslant \omega_{n-1} \leqslant 1$. Каждая гиперплоскость $\sum\limits_{i=1}^{n} C_i^{q_1}\omega_i = \sum\limits_{i=1}^{n} C_i^{q_2}\omega_i$ будет разделять области, в которых будут лучше расписания q_1 или q_2 соответственно. Все C_Q^2 гиперплоскостей различны. Каждому расписанию $q_0 \in$ OPT будет соответствовать некоторый открытый многогранник, в котором это расписание лучше всех остальных. Если такой области для расписания нет, то оно может быть оптимальным при некоторых значениях весов. Однако при этих весах такое же значение целевой функции имеют расписания из OPT, и в результате по совокупности доминируют его. Такие расписания рассматривать не имеет смысла. В программной реализации для формирования OPT использовали неравенства $\sum\limits_{i=1}^{n} C_i^{q_0}\omega_i + \varepsilon \leqslant \sum\limits_{i=1}^{n} C_i^{q}$ для некоторого малого ε.

В табл. 1 приведены значения Q и мощность выделенного подмножества OPT для случая $p = 2$ и $r_j = j - 1$, $j = 1, 2, \ldots, n$.

Таблица 1

n	3	4	5	6	7	8	9	10	11	12	n
Q	4	9	21	51	127	323	835	2188	5798	15511	$\sim e^n$
$\lvert\mathrm{OPT}\rvert$	3	6	12	24	48	96	192	384	768	1532	$3 \cdot 2^{n-3}$

Как видно из табл. 1, мощность этого подмножества также растёт экспоненциально с ростом числа работ. Однако его исследование позволяет получить дополнительные свойства и сформировать подходы к построению алгоритма решения задачи.

Параметрический анализ одного примера при фиксированных r_i, $i = 1, 2, \ldots, n$, и p для $n = 12$ занимает несколько минут на процессоре Intel Core i7-6700HQ 2,60 ГГц. Анализ предусматривает формирование подмножества OPT и поиск весов, при которых расписание $q \in \mathrm{OPT}$ оптимально. С помощью этой программы можно получать граничные значения, в пределах которых могут изменяться веса при сохранении оптимальности расписания q. Для остальных расписаний выдаётся информация, что система строгих неравенств несовместна. При размерности $n > 12$ полный анализ проводить нет необходимости, однако для $n \leqslant 24$ программа позволяет за приемлемое время проводить выборочный анализ конкретных расписаний и проверять различные гипотезы.

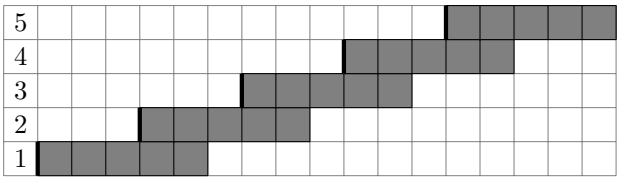


Рис. 3. Входные данные примера

Проиллюстрируем, что может дать такой подход. Как и выше, под каждую работу отводим временну́ю ось, и эти оси располагаем одну под другой, начиная с работы с наибольшим весом, и далее по убыванию. Время поступления работы будем выделять жирной чертой. Рассмотрим пример с пятью работами длительности $p = 5$ и временами поступления $r_1 = 0$, $r_2 = 3$, $r_3 = 6$, $r_4 = 9$, $r_5 = 12$. В нём достаточно отражены свойства, по которым можно проследить основные особенности и сделать необходимые выводы. Входные данные примера изображены на рис. 3. Предобработки входных данных не требуется, так как задача приведённая.

С помощью описанного выше алгоритма построены возможные расписания. Всего их 49, причём оптимальными могут быть только 25. Ниже отражены несколько типичных расписаний, для которых выписанная

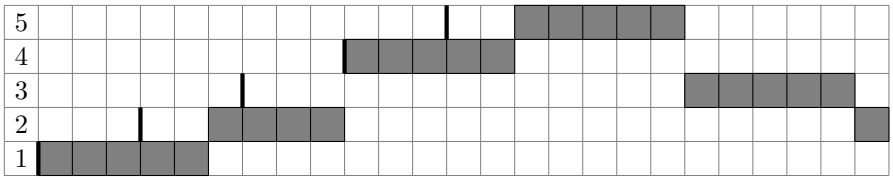


Рис. 4. Прямая перестановка фрагментов расписания

система линейных неравенств несовместна, т. е. решение не будет оптимальным ни при каких весах. Проанализируем причины их неоптимальности.

Самый простой случай — это непосредственная перестановка частей работ. Фрагменты работ 2 и 3 расписания на рис. 4 могут быть переставлены следующим образом (рис. 5.) При этом значение C_2 не изменилось, а C_3 уменьшилось на 3 единицы.

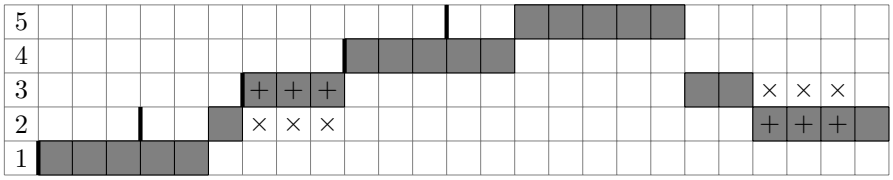


Рис. 5. Исправленное решение

Расписание на рис. 6 также неоптимально по следующей причине. Фрагмент работы 1 уступает фрагменту работы 2, а следовательно, и 3, а далее работе 4 и при этом выигрывает у работы 5, вес которой больше, чем у работы 4: $\frac{5}{\omega_4} < \frac{2}{\omega_3} < \frac{2}{\omega_2} < \frac{2}{\omega_2} < \frac{5}{\omega_5}$. Тем самым $\omega_4 > \omega_5$; противоречие.

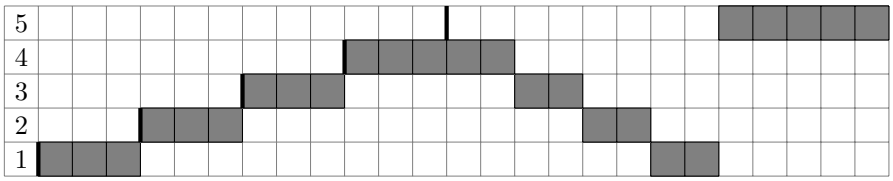


Рис. 6. Транзитивность приоритетов фрагментов работ

С помощью параметрического анализа конкретных примеров удалось выявить и в последствии обосновать несколько свойств оптимальных расписаний.

Утверждение 5. *Для приведённой задачи существует оптимальное расписание, в котором время завершения работы с минимальным весом C_1 кратно p.*

Доказательство очевидно, так как по утверждению 2 все работы, которые прерывали её, должны быть завершены к моменту C_1. Утверждение 5 доказано.

Утверждение 6. *Если в оптимальном решении приведённой задачи выполнено $C_1 \geqslant \max\limits_{j=1,2,\ldots,n} r_j$, то в интервале $[C_1, pn]$ все работы выполняются без прерываний и все фрагменты имеют длительность p.*

Доказательство непосредственно следует из утверждения 2, так как работа 1 имеет наименьший вес. Любая другая работа либо завершится до момента C_1, либо начнёт выполнение после работы 1. Во втором случае условие $C_1 \geqslant \max\limits_{j=1,2,\ldots,n} r_j$ и утверждение 3 гарантируют непрерывность выполнения работ после момента C_1. Утверждение 6 доказано.

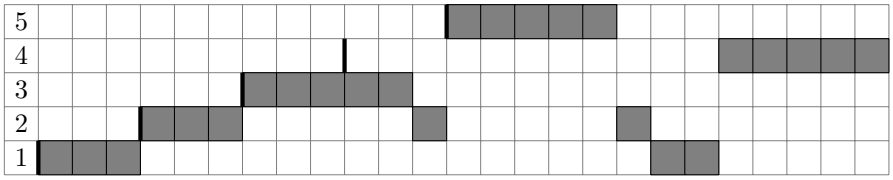


Рис. 7. Неоптимальное расписание

Важным является расписание на рис. 7. Оно не принадлежит множеству OPT, но одного доминирующего его расписания не существует. В данном случае доминирующей является группа из трёх следующих расписаний $(C_1, C_2, C_3, C_4, C_5)$: $(25, 23, 11, 21, 17)$, $(25, 8, 23, 19, 17)$ и $(5, 25, 11, 21, 17)$.

5. Полиномиально разрешимый случай задачи

В данном разделе выделен полиномиально разрешимый случай задачи, когда к моменту поступления очередной работы остаётся невыполненной только одна работа из ранее поступивших. Такое поступление можно назвать разреженным.

Рассмотрим приведённую подзадачу и упорядочим работы по возрастанию времён поступления $r_1 < r_2 < \cdots < r_n$. В приведённой задаче первая работа имеет наименьший вес, $r_1 = 0$ и $r_i < (i-1)p$, $i = 2, 3, \ldots, n$. Пусть для всех $i = 3, 4, \ldots, n$ имеет место условие $r_i \geqslant r_{i-1} + p$. Тогда к моменту r_i успевают завершиться ровно $i-2$ ранее поступивших работ,

и только одна работа к этому моменту не завершает своего выполнения. Пример входных данных такого типа приведён на рис. 8.

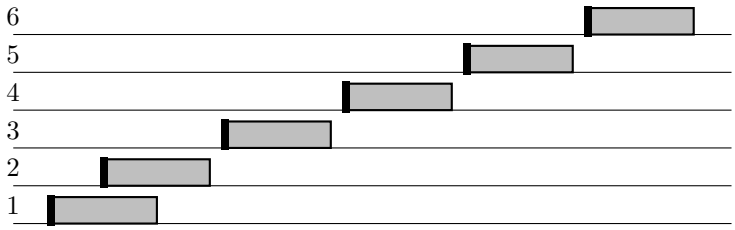


Рис. 8. Пример слабого перекрытия

Ключевым аспектом при составлении расписания является принятие решений в моменты времени r_i, $i = 2, 3, \ldots, n-1$, относительно того, продолжить выполнение текущей работы или прервать её и начать выполнение поступившей работы i. Это справедливо в общем случае. Конкретика для данной подзадачи заключается в том, что к моменту r_i уже $i-2$ работ завершили своё выполнение. Это позволяет реализовать схему динамического программирования с полиномиальной трудоёмкостью.

Обозначим через $\varphi(i, j)$ оптимальное значение взвешенной суммы работ, завершившихся после момента r_i, если в этот момент из ранее начатых не завершена работа $j \in \{1, 2, \ldots, i-1\}$. Такая работа только одна. Необходимо найти $\varphi(2, 1)$.

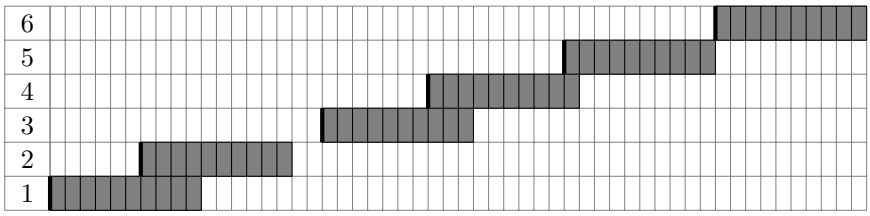


Рис. 9. Входные данные примера

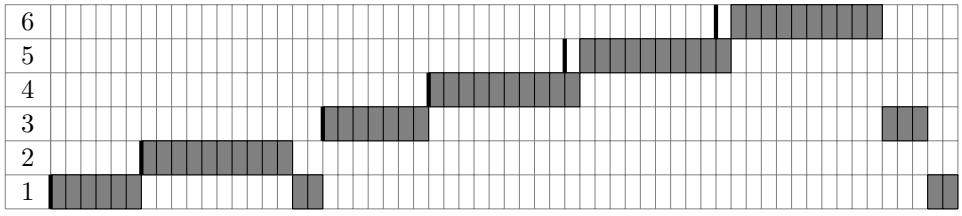


Рис. 10. Оптимальное решение примера

Начинаем с конца, с момента r_n. Для всех $j \in \{1, 2, \ldots, n-1\}$

$$\varphi(n, j) = \min\{(r_n + p)\omega_n + np\omega_j,\ (n-1)p\omega_j + np\omega_n\}.$$

В первом случае приоритет у работы n, а работа j прерывается и завершается в конце. Во втором случае приоритет у работы j, а работа n сдвигается в конец.

Далее для всех $i = n-1, n-2, \ldots, 2$ и всех $j = 1, 2, \ldots, i-1$

$$\varphi(i, j) = \min\{(r_i + p)\omega_i + \varphi(i+1, j),\ (i-1)p\omega_j + \varphi(i+1, i)\}.$$

Построение оптимального решения происходит стандартным способом.

Иллюстративный пример с $n = 6$, $p = 10$, временами поступлений $(0, 6, 10, 25, 34, 44)$ и весами $(8, 12, 18, 30, 33, 50)$ приведён на рис. 9 и 10. Трудоёмкость предложенного алгоритма динамического программирования составляет $O(n^2)$ операций.

Заключение

В статье исследуется задача минимизации среднего взвешенного времени завершения работ одинаковой длительности на одном станке с заданными сроками поступления работ и разрешением прерываний их выполнения. Описан алгоритм предобработки входных данных, позволяющий свести задачу к последовательному решению задач более простой и регулярной структуры. Предложен алгоритм построения конечного подмножества решений, которое содержит оптимальное расписание.

Разработана и реализована технология анализа конкретных расписаний. Это позволяет выдвигать и проверять различные гипотезы по структуре оптимальных расписаний. В частности, для каждого расписания построенного конечного подмножества можно найти веса и границы их изменений, при которых расписание будет оптимальным, или установить, что таких весов нет. Во втором случае можно выделить подмножество расписаний, которое доминирует указанное, и сделать анализ причин этого доминирования. Анализ всех расписаний построенного конечного подмножества позволяет сформировать ещё более узкий класс решений OPT без потери оптимума.

Описан полиномиально разрешимый случай задачи.

Финансирование работы

Исследование выполнено в рамках государственного задания Института математики им. С. Л. Соболева (проект № FWNF–2022–0020). Дополнительных грантов на проведение или руководство этим исследованием получено не было.

Конфликт интересов

Авторы заявляют, что у них нет конфликта интересов.

Литература

1. **Pinedo M. L.** Scheduling: Theory, algorithms, and systems. New York: Springer, 2008. 678 p. DOI: 10.1007/978-0-387-78935-4.
2. **Lenstra J. K., Rinnooy Kan A. H. G., Brucker P.** Complexity of machine scheduling problems // Ann. Discrete Math. 1977. V. 1. P. 343–362. DOI: 10.1016/S0167-5060(08)70743-X.
3. **Baptiste P.** Scheduling equal-length jobs on identical parallel machines // Discrete Appl. Math. 2000. V. 103, No. 1–3. P. 21–32. DOI: 10.1016/S0166-218X(99)00238-3.
4. **Labetoulle J., Lawler E. L., Lenstra J. K., Rinnooy Kan A. H. G.** Preemptive scheduling of uniform machines subject to release dates // Progress in combinatorial optimization. Toronto: Acad. Press, 1984. P. 245–261. DOI: 10.1016/B978-0-12-566780-7.50020-9.
5. **Baker K. R.** Introduction to sequencing and scheduling. New York: John Wiley & Sons, 1974. 318 p.
6. **Баптист Ф., Карлье Ж., Кононов А. В., Керан М., Севастьянов С. В., Свириденко М. И.** Структурные свойства оптимальных расписаний с прерываниями операций // Дискрет. анализ и исслед. операций. 2009. Т. 16, № 1. Р. 3–36.
7. **Batsyn M., Goldengorin B., Pardalos P. M., Sukhov P.** Online heuristic for the preemptive single machine scheduling problem of minimizing the total weighted completion time // Optim. Methods Softw. 2014. V. 29. P. 955–963. DOI: 10.1080/10556788.2013.854360.
8. **Batsyn M., Goldengorin B., Sukhov P., Pardalos P. M.** Lower and upper bounds for the preemptive single machine scheduling problem with equal processing times // Models, algorithms, and technologies for network analysis. Proc. 2nd Int. Conf. Network Analysis (Nizhny Novgorod, Russia, May 7–9, 2012). New York: Springer, 2013. P. 11–27. (Springer Proc. Math. Stat.; V. 59). DOI: 10.1007/978-1-4614-8588-9_2.
9. **Goemans M.-X., Queyranne M., Schulz A.-S., Skutella M., Wang Y.** Single machine scheduling with release dates // SIAM J. Discrete Math. 2002. V. 15, No. 2. P. 165–192. DOI: 10.1137/S089548019936223X.
10. **Kravchenko S. A., Werner F.** Scheduling jobs with equal processing times // IFAC Proc. Volumes. 2009. V. 42, No. 4. P. 1262–1267. DOI: 10.3182/20090603-3-RU-2001.0042.
11. **Fomin A., Goldengorin B.** An efficient model for the preemptive single machine scheduling of equal-length jobs. Ithaca, NY: Cornell Univ., 2020. (Cornell Univ. Libr. e-Print Archive; arXiv:2012.08152). DOI: 10.48550/arXiv.2012.08152.
12. **Chernykh K. A., Servakh V. V.** The structure of the optimal solution of the problem of one machine with the possibility of interruptions of jobs // Proc. 14th Int. Asian School-Seminar «Optimization Problems of Complex Systems» (Kara-Oi, Kyrgyzstan, July 20–31, 2018). Piscataway: IEEE, 2018. P. 312–321.

13. Chernykh K. A., Servakh V. V. Combinatorial structure of optimal solutions to the problem of a single machine with preemption // Proc. 15th Int. Asian School-Seminar «Optimization Problems of Complex Systems» (Novosibirsk, Russia, Aug. 26–30, 2019). Piscataway: IEEE, 2019. P. 21–26. DOI: 10.1109/OPCS.2019.8880148.

14. Chernykh K. A., Servakh V. V. Analysis of optimal solutions to the problem of a single machine with preemption // Mathematical optimization theory and operations research: Recent trends. Rev. Sel. Pap. 20th Int. Conf. (Irkutsk, Russia, July 5–10, 2021). Cham: Springer, 2021. P. 163–174. (Commun. Comput. Inf. Sci.; V. 1476). DOI: 10.1007/978-3-030-86433-0_11.

15. Jaramillo F., Erkoc M. Minimizing total weighted tardiness and overtime costs for single machine preemptive scheduling // Comput. Ind. Eng. 2017. V. 107. P. 109–119. DOI: 10.1016/j.cie.2017.03.012.

16. Jaramillo F., Keles B., Erkoc M. Modeling single machine preemptive scheduling problems for computational efficiency // Ann. Oper. Res. 2020. V. 285. P. 197–222. DOI: 10.1007/s10479-019-03298-9.

Ляшкова Ксения Андреевна
Сервах Владимир Вицентьевич

Статья поступила
26 июля 2022 г.
После доработки —
17 января 2024 г.
Принята к публикации
22 марта 2024 г.

DISKRETNYI ANALIZ I ISSLEDOVANIE OPERATSII
/DISCRETE ANALYSIS AND OPERATIONS RESEARCH/
July–September 2024. Vol. 31, No. 3. P. 105–122

UDC 519.8+518.25 DOI: 10.33048/daio.2024.31.750

THE PROBLEM OF ONE MACHINE WITH EQUAL PROCESSING TIME AND PREEMPTION

K. A. Lyashkova [a] and *V. V. Servakh* [b]

Omsk Branch of the Sobolev Institute of Mathematics,
13 Pevtsov Street, 644099 Omsk, Russia
E-mail: [a] ksech@bk.ru, [b] svv_usa@rambler.ru

Abstract. We consider the problem of minimizing the weighted average execution time of equal-length jobs performance on one machine at the specified time of job arrival and the possibility of their interruption. The computational complexity of this problem is currently unknown. The article proposes an algorithm for preprocessing input data that allows reducing the problem to a narrower and more regular class of examples. The properties of optimal solutions are substantiated. Based on them, an algorithm for constructing a finite subset of solutions containing an optimal schedule has been developed. A parametric analysis of the schedules in this subset has been carried out that makes it possible to form a subclass of schedules that are optimal at some values of weights. A polynomially solvable case of the problem is isolated. Tab. 1, illustr. 10, bibliogr. 16.

Keywords: schedule theory, one machine, preemption.

References

1. **M. L. Pinedo,** *Scheduling: Theory, Algorithms, and Systems* (Springer, New York, 2008), DOI: 10.1007/978-0-387-78935-4.
2. **J. K. Lenstra, A. H. G. Rinnooy Kan,** and **P. Brucker,** Complexity of machine scheduling problems, *Ann. Discrete Math.* **1**, 343–362 (1977), DOI: 10.1016/S0167-5060(08)70743-X.
3. **P. Baptiste,** Scheduling equal-length jobs on identical parallel machines, *Discrete Appl. Math.* **103** (1–3), 21–32 (2000), DOI: 10.1016/S0166-218X(99)00238-3.

English transl.: *Journal of Applied and Industrial Mathematics* **18** (3), 479–488 (2024), DOI: 10.1134/S1990478924030098.

4. **J. Labetoulle, E. L. Lawler, J. K. Lenstra,** and **A. H. G. Rinnooy Kan,** Preemptive scheduling of uniform machines subject to release dates, in *Progress in Combinatorial Optimization* (Acad. Press, Toronto, 1984), pp. 245–261, DOI: 10.1016/B978-0-12-566780-7.50020-9.
5. **K. R. Baker,** Introduction to Sequencing and Scheduling (John Wiley & Sons, New York, 1974).
6. **P. Baptiste, J. Carlier, A. V. Kononov, M. Queyranne, S. V. Sevast'yanov,** and **M. I. Sviridenko,** Structural properties of optimal schedules with preemption, *Diskretn. Anal. Issled. Oper.* **16** (1), 3–36 (2009) [Russian] [*J. Appl. Ind. Math.* **4** (4), 455–474 (2010)].
7. **M. Batsyn, B. Goldengorin, P. M. Pardalos,** and **P. Sukhov,** Online heuristic for the preemptive single machine scheduling problem of minimizing the total weighted completion time, *Optim. Methods Softw.* **29**, 955–963 (2014), DOI: 10.1080/10556788.2013.854360.
8. **M. Batsyn, B. Goldengorin, P. Sukhov,** and **P. M. Pardalos,** Lower and upper bounds for the preemptive single machine scheduling problem with equal processing times, in *Models, Algorithms, and Technologies for Network Analysis* (Proc. 2nd Int. Conf. Network Analysis, Nizhny Novgorod, Russia, May 7–9, 2012) (Springer, New York, 2013), pp. 11–27 (Springer Proc. Math. Stat., Vol. 59), DOI: 10.1007/978-1-4614-8588-9_2.
9. **M.-X. Goemans, M. Queyranne, A.-S. Schulz, M. Skutella,** and **Y. Wang,** Single machine scheduling with release dates, *SIAM J. Discrete Math.* **15** (2), 165–192 (2002), DOI: 10.1137/S089548019936223X.
10. **S. A. Kravchenko** and **F. Werner,** Scheduling jobs with equal processing times, *IFAC Proc. Volumes* **42** (4), 1262–1267 (2009), DOI: 10.3182/20090603-3-RU-2001.0042.
11. **A. Fomin** and **B. Goldengorin,** An efficient model for the preemptive single machine scheduling of equal-length jobs (Cornell Univ., Ithaca, NY, 2020) (Cornell Univ. Libr. e-Print Archive, arXiv:2012.08152), DOI: 10.48550/arXiv.2012.08152.
12. **K. A. Chernykh** and **V. V. Servakh,** The structure of the optimal solution of the problem of one machine with the possibility of interruptions of jobs, in *Proc. 14th Int. Asian School-Seminar "Optimization Problems of Complex Systems", Kara-Oi, Kyrgyzstan, July 20–31, 2018* (IEEE, Piscataway, 2018), pp. 312–321.
13. **K. A. Chernykh** and **V. V. Servakh,** Combinatorial structure of optimal solutions to the problem of a single machine with preemption, in *Proc. 15th Int. Asian School-Seminar "Optimization Problems of Complex Systems", Novosibirsk, Russia, Aug. 26–30, 2019* (IEEE, Piscataway, 2019), pp. 21–26, DOI: 10.1109/OPCS.2019.8880148.
14. **K. A. Chernykh** and **V. V. Servakh,** Analysis of optimal solutions to the problem of a single machine with preemption, in *Mathematical Optimization Theory and Operations Research: Recent Trends* (Rev. Sel. Pap. 20th Int. Conf., Irkutsk, Russia, July 5–10, 2021) (Springer, Cham, 2021), pp. 163–174 (Commun. Comput. Inf. Sci., Vol. 1476), DOI: 10.1007/978-3-030-86433-0_11.

15. F. Jaramillo and **M. Erkoc,** Minimizing total weighted tardiness and overtime costs for single machine preemptive scheduling, *Comput. Ind. Eng.* **107**, 109–119 (2017), DOI: 10.1016/j.cie.2017.03.012.

16. F. Jaramillo, B. Keles, and **M. Erkoc,** Modeling single machine preemptive scheduling problems for computational efficiency, *Ann. Oper. Res.* **285**, 197–222 (2020), DOI: 10.1007/s10479-019-03298-9.

Ksenia A. Lyashkova
Vladimir V. Servakh

Received July 26, 2022
Revised January 17, 2024
Accepted March 22, 2024