УДК 519.8
MSC 90B35

Алгоритм составления расписания для одного процессора с гарантированной оценкой точности 3/2

Н. С. Григорьева

Санкт-Петербургский государственный университет, Российская Федерация,
199034, Санкт-Петербург, Университетская наб., 7–9

Для цитирования: *Григорьева Н. С.* Алгоритм составления расписания для одного процессора с гарантированной оценкой точности 3/2 // Вестник Санкт-Петербургского университета. Прикладная математика. Информатика. Процессы управления. 2021. Т. 17. Вып. 3. С. 240–253. https://doi.org/10.21638/11701/spbu10.2021.302

Рассматривается задача составления расписания для одного процессора $1|r_i, q_i|C_{\max}$, в которой для каждого задания известны времена поступления, времена выполнения и времена доставки. Предлагается новый приближенный алгоритм решения задачи $1|r_i, q_i|C_{\max}$ с гарантированной оценкой точности 3/2 и вычислительной сложностью $O(n \log n)$. Приводятся пример, показывающий, что данная оценка асимптотически достигается, и результаты вычислительного эксперимента, свидетельствующие о быстродействии и практической точности алгоритма.

Ключевые слова: задача составления расписания, времена поступления, времена доставки, приближенный алгоритм, гарантированная оценка точности.

1. Введение. В работе рассматривается задача составления расписания, в которой n заданий выполняется на одном процессоре. Пусть U — множество заданий и для каждого задания $i \in U$ заданы: время поступления на исполнение — $r(i)$, время выполнения — $t(i)$ и время доставки — $q(i)$.

Считаем, что доставка начинается сразу после завершения выполнения задания процессором и может осуществляться одновременно с его работой. Прерывания выполнения заданий не допускаются, и в каждый момент времени процессор может исполнять не более одного задания. Требуется построить расписание, т. е. найти для каждого задания i время начала его выполнения $\tau(i)$, при условии, что $r(i) \leqslant \tau(i)$. Характеристика качества расписания — время доставки последней работы $C_{\max} = \max\{\tau(i) + t(i) + q(i) | i \in U\}$. Требуется построить расписание, минимизирующее $C_{\max}$.

Следуя схеме, предложенной Р. Грехемом и др. [1], задача обозначается как $1|r_i, q_i|C_{\max}$. В [2] показано, что задача NP-трудная в сильном смысле, но существуют точные полиномиальные алгоритмы для ряда специальных случаев. В [3] рассматривалась эквивалентная постановка задачи, в которой вместо времени доставки для каждой работы известен директивный срок $D(i) = K - q(i)$, где K — константа, и целевой функцией является максимальное временное смещение $L_{\max} = \max\{\tau(i) + t(i) - D(i) | i \in U\}$. Эта постановка задачи обозначается как $1|r_i|L_{\max}$. Преимущество модели с временами доставки заключается в том, что целевая функция всегда положительна, тогда как максимальное временное смещение может быть отрицательным или равным нулю. Если поменять местами времена доставки и времена поступления заданий, получим инверсную задачу со свойством, что решение пря-

мой задачи $S = (i_1, i_2, ..., i_n)$ оптимально тогда и только тогда, когда перестановка $S_{inv} = (i_n, i_{n-1}, ..., i_1)$ есть оптимальное решение инверсной задачи.

В большинстве исследований задач составления расписаний рассматриваются незадерживающие расписания, определенные К. Бейкером [4] как допустимые расписания, в которых процессор не должен простаивать, если есть задания, которые процессор может выполнять. Одним из популярных инструментов построения расписаний служат списочные алгоритмы, которые строят незадерживающие расписания. В списочном алгоритме на каждом шаге из множества готовых заданий выбирается задание с максимальным приоритетом. Но оптимальное расписание может не принадлежать классу незадерживающих расписаний.

Расписания с невынужденными простоями (inserted idle time — IIT) были определены в [5] как допустимые расписания, в которых процессор может простаивать при наличии готовых к выполнению заданий. В [5] авторы приводят обзор литературы моделей составления расписаний, в которых актуально использование невынужденных простоев.

Приближенные расписания с невынужденными простоями для параллельных процессоров рассматривались: для задачи $P|\text{prec}|C_{\max}$, в которой на множестве заданий известно отношение частичного порядка [6], и для задачи $P|r_j, q_j|C_{\max}$ с заданными временами поступления и доставки [7, 8]. Основной идеей алгоритмов для решения этих задач был выбор на каждом шаге самого приоритетного задания, до начала выполнения которого процессор мог простаивать.

В [9] для задачи с одним процессором был предложен допускающий невынужденные простои алгоритм с гарантированной оценкой точности, равной 2. В настоящей работе предлагается новый приближенный алгоритм составления расписания для задачи с одним процессором.

Известны несколько алгоритмов с гарантированной оценкой точности для решения задачи $1|r_i, q_i|C_{\max}$.

Первым алгоритмом построения приближенного расписания была эвристика Шраге — расширенное правило Джексона, которое формулируется следующим образом: каждый раз, когда процессор освобождается, на него назначается готовая работа с максимальным временем доставки [10]. Вычислительная сложность алгоритма $O(n \log n)$. В работе Х. Кизе и др. [11] показано, что алгоритм имеет гарантированную оценку точности, равную 2.

К. Поттс [12] предложил алгоритм, в котором n раз запускается процедура, реализующая правило Джексона. Перед каждым новым запуском к ограничениям задачи добавляется новое ограничение на порядок выполнения заданий. Из n построенных расписаний выбирается лучшее. Вычислительная сложность алгоритма Поттса $O(n^2 \log n)$. Гарантированная оценка точности равна $3/2$.

Л. Холл и Д. Шмойс [13] рассматривали наряду с прямой задачей инверсную задачу, в которой времена поступления заданий и времена доставки меняются местами. Авторы разработали метод построения расписания, в котором алгоритм Поттса применяется для прямой и инверсной задач. Отдельно рассматривается случай, в котором есть два длительных задания. В общей сложности алгоритм строит $4n$ расписаний и выбирает из них лучшее. Вычислительная сложность алгоритма $O(n^2 \log n)$. Гарантированная оценка точности равна $4/3$.

Е. Новицкий и К. Смутницкий [14] предложили алгоритм с гарантированной оценкой точности $3/2$, который создает только две перестановки. Первый раз применяется правило Джексона, далее определяется интерференционная работа, и все

множество заданий разбивается на два множества: задания, которые должны выполняться до интерференционной работы, и задания, которые должны выполняться после нее. Задания из первого множества устанавливаются в порядке поступления, из второго — по невозрастанию времени доставки. Выбирается лучшее расписание из двух построенных. Вычислительная сложность $O(n \log n)$. Гарантированная оценка точности равна $3/2$.

Все упомянутые алгоритмы используют списочный жадный алгоритм Шраге как базовую эвристику.

Для задачи $1|r_i, q_i|C_{\max}$ известны также полиномиально-временные аппроксимационные схемы (PTAS). Две такие схемы предложили Л. Холл и Д. Шмойс [13, 15], а М. Мастролилли [16] улучшил их результаты, предложив PTAS с линейной сложностью.

Е. Новицкий и К. Смутницкий [14] поставили в своей работе вопрос: какое минимальное количество перестановок надо построить, чтобы гарантировать точность алгоритма в $3/2$ или $4/3$ при условии, что получение каждой перестановки требует не более чем $O(n \log n)$ операций? Для получения точности $3/2$ они предложили алгоритм, который строит две перестановки. Вопрос о том, сколько перестановок необходимо, чтобы точность алгоритма была равна $4/3$, остается неизученным.

В настоящей работе предлагается приближенный алгоритм решения задачи с гарантированной оценкой точности $3/2$, который создает две перестановки: одну — методом Шраге, а вторую — алгоритмом с невынужденными простоями IJR. Построение каждой перестановки требует $O(n \log n)$ действий. Алгоритм IJR является жадным, но не списочным, и его можно использовать как базовую эвристику при составлении расписаний для различных моделей и конструирования метода ветвей и границ.

В списочных расписаниях сначала определяется множество готовых заданий, а затем из готовых выбирается задание с максимальным приоритетом, для которого приоритет — это время доставки. В предложенном алгоритме выбираются два задания: самое приоритетное задание и самое приоритетное из готовых. Затем принимается решение, какое из двух заданий выполнять.

Основная идея алгоритма IJR состоит в следующем: иногда лучше поставить на обслуживание приоритетное задание, даже если это приведет к простою процессора, чем загружать процессор менее приоритетным. Установлены дополнительные условия, в которых выгодно организовать невынужденный простой процессора. Такие условия позволяют сделать выбор между двумя заданиями.

Статья организована следующим образом. В п. 2 излагаются новый приближенный алгоритм построения расписания с невынужденными простоями IJR и комбинированный алгоритм ICA, который строит две перестановки и выбирает из них лучшую. В п. 3 изучаются свойства построенного расписания и доказывается гарантированная оценка точности алгоритма ICA, равная $3/2$. Рассматривается пример, показывающий, что данная оценка асимптотически достигается. В п. 4 приведены результаты вычислительного эксперимента. В п. 5 формулируются основные результаты, полученные в статье.

2. Алгоритмы построения расписаний IJR и ICA. Сначала опишем алгоритм построения расписания с невынужденными простоями IJR.

Введем следующие обозначения: $r_{\min} = \min\{r(i) | \ i \in U\}$, $q_{\min} = \min\{q(i) \mid i \in U\}$. Определим нижнюю границу целевой функции LB по формуле

$$LB = \max\left\{r_{\min} + \sum_{i=1}^{n} t(i) + q_{\min}, \max\{r(i) + t(i) + q(i) \mid i \in U\}\right\}.$$

Отсортируем задания по неубыванию времен поступления: $r(j_1) \leqslant r(j_2) \leqslant \ldots \leqslant r(j_n)$. Готовые задания будем хранить в очереди с приоритетами Q_1, приоритетом является время доставки.

Пусть $time$ — время освобождения процессора после выполнения уже поставленных в расписание заданий, в начальный момент времени положим $time := r_{\min}$.

Пусть $k-1$ задание уже поставлено в расписание, S_{k-1} — подмножество заданий, включенных в расписание. Известно время освобождения процессора: $time := \max\{\tau(i) + t(i) | i \in S_{k-1}\}$. Опишем k-й шаг.

1. Добавляем в очередь Q_1 готовые задания такие, что $r(i) \leqslant time$.

2. Если очередь Q_1 пуста, то $time := \min\{r(i) | i \notin S_{k-1}\}$ и переходим к п. 1.

3. Выбираем готовое задание $u \in Q_1$ с максимальным временем доставки $q(u) = \max\{q(i) | i \in Q_1\}$.

4. Положим $r_{up} := time + t(u)$.

5. Ищем задание u^*, конкурирующее с заданием u. Для этого находим первое по порядку, еще не включенное в очередь Q_1 задание j_i такое, что $time < r(j_i) < r_{up}$. Если такого задания нет, переходим к п. 9.

6. Если выполнено $q(j_i) > LB/2$, то для задания j_i определяем возможный простой процессора $\mathrm{idle}(j_i) = r(j_i) - time$. Проверяем условие $q(j_i) - q(u) \geqslant \mathrm{idle}(j_i)$.

7. Если условия п. 6 выполнены, то устанавливаем на процессор задание $u^* = j_i$, положим $\tau(j_i) := r(j_i)$; $time := \tau(j_i) + t(j_i)$; переходим к п. 1. Иначе добавляем задание j_i в очередь.

8. Если $r(j_{i+1}) < r_{up}$, то положим $i := i + 1$, перейдем к п. 6.

9. Если не нашли задание u^*, то устанавливаем на процессор задание u, положим $\tau(u) := time$; $time := \tau(u) + t(u)$.

10. Если $k < n$, то $k := k + 1$, затем переходим к п. 1.

11. Конец алгоритма, расписание S_n построено.

Находим значение целевой функции $C_{\max}(S_n) = \max\{\tau(i) + t(i) + q(i) \mid i \in U\}$.

Опишем комбинированный алгоритм, который строит два расписания и выбирает из них лучшее.

Комбинированный алгоритм построения расписания ICA:

1. Строим расписание S_{JR} алгоритмом JR, обозначим длину расписания $C_{\max}(S_{JR})$.

2. Строим расписание S алгоритмом IJR, обозначим длину расписания $C_{\max}(S)$.

3. Выбираем расписание S_A с меньшим значением целевой функции: $C_{\max}(S_A) = \min\{C_{\max}(S), C_{\max}(S_{JR})\}$.

3. Свойства расписания, построенного алгоритмом ICA. Рассмотрим свойства построенного расписания. Пусть с помощью алгоритма IJR сгенерировано расписание S, для каждого задания j_i определено время начала выполнения задания $\tau(j_i)$. Значение целевой функции равно $C_{\max}(S)$. Рассмотрим ряд определений, которые были введены в [12] для расписаний, построенных по правилу Джексона и являющихся важными характеристиками при изучении расписания с простоями.

Определение 1. *Критической* работой называется работа j_c такая, что $C_{\max}(S) = \tau(j_c) + t(j_c) + q(j_c)$. Если таких работ несколько, то выберем самую раннюю в расписании S.

Определение 2. *Критической* последовательностью в расписании S называется последовательность работ $J(S) = (j_a, j_{a+1}, \ldots, j_c)$ такая, что j_c — критическая работа и в расписании нет простоя процессора, начиная с момента начала работы j_a до окончания работы j_c.

Работа j_a либо является самой первой в расписании, либо в расписании перед ее началом процессор простаивает.

Определение 3. Работа j_u в критической последовательности называется *интерференционной*, если $q(j_u) < q(j_c)$ и $q(j_i) \geqslant q(j_c)$ для $i > u$.

Утверждение [12]**.** *Если для всех работ критической последовательности верно, что* $r(j_i) \geqslant r(j_a)$ *и* $q(j_i) \geqslant q(j_c)$, *то расписание оптимально.*

Введем определение задержанной работы, которая может встретиться в расписаниях с простоями.

Определение 4. Работу j_v из критической последовательности будем называть *задержанной*, если $r(j_v) < r(j_a)$.

Интерференционная работа может быть задержанной.

Сформулируем и докажем два свойства расписания, аналогичные свойствам расписаний, построенных по правилу Джексона [12].

Лемма 1. *Если в критической последовательности есть интерференционная работа* j_u, *то* $C_{\max}(S) - C_{\max}(S_{\text{opt}}) \leqslant t(j_u) - \text{idle}$, *где* $\text{idle} > 0$ *— возможный простой в расписании.*

Доказательство. Обозначим общее время выполнения работ критической последовательности через $T(J(S)) = \sum_{i=a}^{c} t(j_i)$, тогда

$$C_{\max}(S) = r(j_a) + T(J(S)) + q(j_c).$$

Если в критической последовательности есть интерференционная работа j_u, то эту последовательность можно представить как $J(S) = (S_1, j_u, S_2)$, где S_1 — последовательность работ до работы j_u, а S_2 — последовательность работ после j_u.

Получим, что в момент времени $t_1 = r(j_a) + T(S_1)$ ни одна из работ последовательности S_2 не готова к выполнению. Введем следующее обозначение: $r_{\min}(S_2) = \min\{r(j_i) | j_i \in S_2\}$. Тогда выполнено $t_1 < r_{\min}(S_2)$.

Для оптимального расписания верно

$$C_{\max}(S_{\text{opt}}) \geqslant r_{\min}(S_2) + T(S_2) + q(j_c),$$

откуда

$$C_{\max}(S) - C_{\max}(S_{\text{opt}}) \leqslant r(j_a) + T(J(S)) + q(j_c) - r_{\min}(S_2) - T(S_2) - q(j_c) =$$

$$= r(j_a) + T(S_1) + t(j_u) - r_{\min}(S_2) = t(j_u) - \text{idle},$$

где $\text{idle} = r_{\min}(S_2) - t_1$ — минимальный простой процессора, если вместо интерференционной работы поставить в расписание работу из последовательности S_2. □

Эта лемма уточняет свойство расписания Джексона, доказанное в [12], и показывает, что, убрав интерференционную работу, можно получить выигрыш не больше, чем $t(j_u)$ — idle.

Лемма 2. *Если в критической последовательности нет задержанных работ, то* $C_{\max}(S) - C_{\max}(S_{\text{opt}}) \leqslant q(j_c)$.

Лемма сформулирована и доказана Поттсом [12], для рассматриваемого алгоритма необходимо лишь добавить уточнение об отсутствии задержанных работ.

Пусть с помощью алгоритма IJR построено расписание S, значение целевой функции для которого равно $C_{\max}(S)$, критическая последовательность в расписании $J(S) = (j_a, j_{a+1}, \ldots, j_c)$. А с помошью алгоритма JR построено расписание S_{JR}, в котором критическая последовательность $J(S_{JR}) = (z_a, z_{a+1}, \ldots, z_c)$.

Лемма 3. *Пусть в критической последовательности $J(S) = (S_1, j_u, S_2)$ есть интерференционная работа j_u. Если в оптимальном расписании работа j_u выполняется после последовательности S_2, то верно $C_{\max}(S)/C_{\max}(S_{\text{opt}}) \leqslant 3/2$.*

Доказательство. Если в оптимальном расписании работа j_u выполняется после последовательности S_2, то верно $C_{\max}(S_{\text{opt}}) \geqslant r_{\min}(S_2) + T(S_2) + t(j_u) + q(j_u)$.

Тогда

$$C_{\max}(S) - C_{\max}(S_{\text{opt}}) \leqslant r(j_a) + T(J(S)) + q(j_c) - r_{\min}(S_2) - T(S_2) - t(j_u) - q(j_u) =$$

$$= \text{idle} + q(j_c) - q(j_u).$$

Выберем работу $v \in S_2$ такую, что $r(v) = r_{\min}(S_2)$.

Тогда верно, что либо $\text{idle} = \text{idle}(v) = r_{\min}(S_2) - r(j_a) - T(S_1) > q(v) - q(j_u) \geqslant q(j_c) - q(j_u)$, либо $q(c) \leqslant q(v) < LB/2$. □

Лемма 4. *Пусть в критической последовательности $J(S_{JR}) = (F_1, j_u, F_2)$ есть интерференционная работа j_u. Тогда если в оптимальном расписании работа j_u выполняется до последовательности F_2, то верно $C_{\max}(S_{JR})/C_{\max}(S_{\text{opt}}) \leqslant 3/2$.*

Доказательство. Если $t(j_u) \leqslant C_{\max}(S_{\text{opt}})/2$, то утверждение леммы верно. Пусть $t(j_u) > C_{\max}(S_{\text{opt}})/2$. Если в оптимальном расписании j_u выполняется до всех работ последовательности F_2, то

$$C_{\max}(S_{\text{opt}}) \geqslant r(z_a) + t(j_u) + T(F_2) + q(z_c).$$

Тогда $C_{\max}(S_{JR}) - C_{\max}(S_{\text{opt}}) \leqslant T(F_1) < LB/2$. □

Теорема. *С помощью алгоритма строится расписание S_A, для которого $C_{\max}(S_A)/C_{\max}\ (S_{\text{opt}}) \leqslant 3/2$. Вычислительная сложность алгоритма $O(n \log n)$.*

Доказательство. Пусть с помощью алгоритма IJR построено расписание S, значение целевой функции которого равно $C_{\max}(S)$, критическая последовательность $J(S) = (j_a, j_{a+1}, \ldots, j_c)$; с помощью алгоритма JR — расписание S_{JR}, критическая последовательность $J(S_{JR}) = (z_a, z_{a+1}, \ldots, z_c)$.

Рассмотрим следующие варианты.

1. В одной из построенных критических последовательностей нет интерференционных и задержанных работ. Здесь соответствующий алгоритм построил оптимальное расписание.

2. В каждой критической последовательности нашлась интерференционная работа j_u. Требуется рассмотреть случай, в котором это одна и та же большая работа такая, что $t(j_u) > C_{\max}(S_{\text{opt}})/2$.

3. В критической последовательности $J(S_{JR}) = (z_a, z_{a+1}, \ldots, z_c)$ есть интерференционная работа, а в $J(S) = (j_a, j_{a+1}, \ldots, j_c)$ — задержанные работы.

Разберем варианты 2 и 3. В варианте 2 длина расписания S_{JR} равна $C_{\max}(S_{JR}) = r(z_a) + T(J(S_{JR})) + q(z_c)$.

Если $t(j_u) \leqslant C_{\max}(S_{\text{opt}})/2$, то по лемме 1 утверждение теоремы выполнено. Пусть далее $t(j_u) > C_{\max}(S_{\text{opt}})/2$. Если время доставки критической работы z_c не превосходит $C_{\max}(S_{\text{opt}})/2$, то по лемме 2 утверждение теоремы выполнено. Пусть $q(z_c) > C_{\max}(S_{\text{opt}})/2$.

В силу доказанных лемм 3 и 4 достаточно рассмотреть случай, в котором в оптимальном расписании работа j_u должна стоять после всех работ последовательности F_2 и до работ последовательности S_2. Это может быть только, если $r_{\min}(S_2) > r(z_c) + t(z_c)$.

Рассмотрим расписание S, построенное алгоритмом IJR, его длина равна $C_{\max}(S) = r(j_a) + T(J(S)) + q(j_c)$.

Покажем, что если работа j_u стоит в S до всех работ последовательности F_2, то оценка точности алгоритма IJR равна $3/2$. Пусть работа j_u стоит в S до всех работ последовательности F_2.

Так как $t(j_u) > C_{\max}(S_{\text{opt}})/2$, а $q(j) > C_{\max}(S_{\text{opt}})/2$ для $j \in F_2$ и $r(j) < C_{\max}(S_{\text{opt}})/2$, то все работы последовательности F_2 являются конкурирующими для j_u и должны входить в последовательность S_2 критической последовательности $J(S)$. В расписании S работа j_u может стоять до всех работ последовательности F_2, только если потенциальный простой $\text{idle}(u^*)$ перед конкурирующей работой u^* велик и верно неравенство $q(u^*) - q(j_u) < \text{idle}(u^*)$.

Выберем работу $v \in S_2$ такую, что $r(v) = r_{\min}(S_2) = r_{\min}(F_2)$. Тогда верно, что $\text{idle} = \text{idle}(v) = r_{\min}(F_2) - r(j_a) - T(F_1) > q(v) - q(j_u)$. Тогда по лемме 1

$$C_{\max}(S) - C_{\max}(S_{\text{opt}}) \leqslant t(j_u) - \text{idle}.$$

Покажем, что в этом случае гарантированная оценка точности алгоритма IJR равна $3/2$. Предположим противное, что

$$C_{\max}(S_{\text{opt}})/2 < C_{\max}(S) - C_{\max}(S_{\text{opt}}) \leqslant t(j_u) - \text{idle},$$

тогда

$$C_{\max}(S_{\text{opt}})/2 < t(j_u) - \text{idle}.$$

Следовательно, $q(v) - q(j_u) + C_{\max}(S_{\text{opt}})/2 < t(j_u)$, откуда

$$C_{\max}(S_{\text{opt}}) < t(j_u) + q(j_u) < LB.$$

Получили противоречие, следовательно, если алгоритм IJR поставил работу j_u до F_2, то оценка точности алгоритма равна $3/2$.

Если алгоритм IJR поставил работу j_u после последовательности F_2, то возможны два случая.

Первый случай: в расписании S нет простоев после выполнения работы z_c и до конца критической последовательности работы j_c, тогда в оптимальном расписании работа j_u выполняется между j_a и j_c.

Если в оптимальном расписании работа j_u выполняется между работами j_a и j_c, тогда

$$C_{\max}(S_{\text{opt}}) \geqslant r(j_a) + t(j_u) + T(S_2) + q(j_c).$$

Таким образом,

$$C_{\max}(S) - C_{\max}(S_{\text{opt}}) \leqslant r(j_a) + T(S_1) + t(j_u) + T(S_2) + q(j_c) -$$

$$- r(j_a) - t(j_u) - T(S_2) - q(j_c) = T(S_1) \leqslant LB/2.$$

В этом случае алгоритм IJR построит расписание с оценкой $3/2$.

Второй случай: $r(j_a) > \max\{\tau(z_c) + t(z_c), r_{\min}(F_2) + T(F_2)\}$. По свойствам алгоритма IJR процессор простаивает до момента времени $r(j_a)$ и $q(j_a) > LB/2$.

Тогда в оптимальном расписании работа j_u может выполняться между работами j_a и j_c или до работы j_a.

Осталось рассмотреть случай, когда в оптимальном расписании j_u выполняется после работы z_c и до работы j_a. Тогда $C_{\max}(S_{\text{opt}}) \geqslant r_{\min}(F_2) + T(F_2) + t(j_u) + t(j_a) + q(j_a)$.

Следовательно,

$$C_{\max}(S_{JR}) - C_{\max}(S_{\text{opt}}) \leqslant r(z_a) + T(J(S_{JR})) + q(z_c) -$$

$$- r_{\min}(F_2) - T(F_2) - t(j_u) - t(j_a) - q(j_a) =$$

$$= r(z_a) + T(F_1) + q(z_c) - r_{\min}(F_2) - t(j_a) - q(j_a) <$$

$$< q(z_c) - q(j_a) < LB/2.$$

Это верно, так как $q(z_c) < LB$ и $q(j_a) > LB/2$. В этом случае алгоритм Шраге построит расписание с оценкой, равной $3/2$.

В варианте 3, если в критической последовательности $J(S)$ нет интерференционной работы, то $q(j_i) \geqslant q(j_c)$ для всех работ из критической последовательности $j_i \in J(S)$. Но в критической последовательности есть работы, которые можно начать раньше первой работы j_a. Пусть $r(J(S)) = \min\{r(j_i) \mid j_i \in J(S)\}$.

Тогда

$$C_{\max}(S_{\text{opt}}) \geqslant r(J(S)) + T(J(S)) + q(j_c),$$

$$C_{\max}(S) - C_{\max}(S_{\text{opt}}) \leqslant r(j_a) + T(J(S)) + q(j_c) - r(J(S)) - T(J(S)) - q(j_c) =$$

$$= r(j_a) - r(J(S)) < LB/2.$$

По свойствам алгоритма $r(j_a) < LB/2$.

Гарантированная оценка точности доказана.

Рассмотрим трудоемкость алгоритма построения расписания. Алгоритм строит две перестановки: одну — алгоритмом JR, вычислительная сложность которого $O(n \log n)$, вторую — алгоритмом IJR. Покажем, что для алгоритма IJR вычислительная сложность $O(n \log n)$. Сначала задания сортируются по неубыванию времен поступления: $r(j_1) \leqslant r(j_2) \leqslant \ldots \leqslant r(j_n)$, этот шаг требует $O(n \log n)$ действий. Основной операцией является выбор задания из множества готовых заданий в момент времени $time$. Готовые задания будем хранить, как очередь с приоритетами Q_1, которую можно организовать в виде двоичной кучи, приоритетом будет время доставки $q(j)$.

На шаге 1 алгоритма добавляем в очередь новые готовые задания такие, что $r(j_i) \leqslant time$, добавление каждого требует $O(\log n)$ действий. Задание u с максимальным приоритетом выбирается за $O(1)$ действий. Полагаем, что $r_{up} = time + t(u)$.

На шагах 5–8 добавляем в очередь новые готовые задания, для которых $r(j_i) < r_{up}$. Если нашлось задание, для которого выполнены условия п. 6, то устанавливаем его на процессор. Иначе — просматриваем всех кандидатов, помещая их в очередь, и ставим на процессор задание u.

Для постановки в расписание выбирается задание u^* или u, что требует $O(1)$ действий, и поставленное задание удаляется из очереди Q_1, что требует $O(\log n)$ действий.

Каждое задание может быть добавлено в очередь не более одного раза: построение двоичной кучи требует $O(n \log n)$ действий. Общая вычислительная сложность $O(n \log n)$. □

Приведем пример, в котором оценка достигается.

Лемма 5. *Существует пример, для которого отношение* $C_{\max}(S_A)/C_{\max}(S_{\text{opt}})$ *стремится к 3/2.*

Доказательство. Рассмотрим систему из следующих заданий x, a, u, c. Данные для системы заданий приведены в табл. 1, где M — константа.

Таблица 1. **Данные для системы заданий**

Job	r_i	t_i	q_i
x	ε	ε	$M-2\varepsilon$
a	$M/2-\varepsilon$	ε	$M/2$
u	0	$M/2+\varepsilon$	0
c	$M/2+\varepsilon$	ε	$M/2-2\varepsilon$

Нижняя оценка целевой функции $LB = M$. Алгоритм IJR построит расписание $S = (x, a, u, c)$. Перед началом выполнения задания a процессор будет простаивать $M/2 - \varepsilon$ единиц времени.

Значение целевой функции $C_{\max}(S) = M/2-\varepsilon+\varepsilon+M/2+\varepsilon+\varepsilon+M/2-2\varepsilon = 3/2M$. Алгоритм JR построит расписание $S_{JR} = (u, x, a, c)$. Значение целевой функции $C_{\max}(S_{JR}) = M/2 + \varepsilon + M - 2\varepsilon = 3/2M - \varepsilon$.

Оптимальное расписание $S_{\text{opt}} = (x, u, a, c)$, значение целевой функции для которого $C_{\max}(S_{\text{opt}}) = 2\varepsilon + M/2 + \varepsilon + M/2 - 2\varepsilon = M + \varepsilon$. При ε, стремящимся к нулю, отношение $C_{\max}(S_A)/C_{\max}(S_{\text{opt}})$ стремится к $3/2$. □

4. Вычислительный эксперимент. Для выяснения практической эффективности алгоритма был проведен вычислительный эксперимент. Его целями были:

— программная реализация предложенного алгоритма;

— проверка быстродействия и эффективности алгоритма и экспериментальная оценка его точности на случайных тестовых примерах;

— сравнение точности алгоритма IJR с точностью алгоритма JR. Сравнение точности комбинированного алгоритма ICA с точностью алгоритма NS.

Исходные данные генерировались методом, описанным Дж. Карлье [17]. Такой же метод генерирования тестовых примеров использовали Е. Новицкий и К. Смутницкий при сравнении предложенного ими алгоритма с алгоритмами Холл, Шмойса и Шраге. Для каждого задания с равной вероятностью выбираются целочисленные значения $q(i)$ из диапазона от 1 до $q_{\max}$, значения $r(i)$ из диапазона от 1 до $r_{\max}$ и $t(i)$ из диапазона от 1 до $t_{\max}$.

Были установлены следующие значения: $t_{\max} = 50$, $r_{\max} = q_{\max} = nK$. Были рассмотрены примеры при K от 10 до 22, которые были отмечены Карлье как наиболее трудные для рассматриваемой задачи. Для каждого значения n и K строилось расписание для 100 тестов. Были рассмотрены три группы примеров. Времена выполнения заданий для каждой из групп выбирались из следующих интервалов:

1) $t(j)$ из $[1, t_{\max}]$;

2) $t(j)$ из $[1, t_{\max}/2]$ для $j \in 1 : n-1$ и $t(j_n)$ из $[nt_{\max}/8, 3nt_{\max}/8]$;

3) $t(j)$ из $[1, t_{\max}/3]$ для $j \in 1 : n-2$ и $t(j_{n-1}), t(j_n)$ из $[nt_{\max}/12, 3nt_{\max}/12]$.

Группы 2 и 3 содержат примеры с одним и двумя длительными заданиями. Величина целевой функции $C_{\max}$ сравнивалась с оптимальным значением целевой функции C_{opt}, которое было получено методом ветвей и границ. В табл. 2–5 n — количество заданий в тесте. В табл. 2 оно изменялось от 50 до 5000 и для всех тестов было выбрано $K = 20$.

Результаты работы приближенных алгоритмов IJR, JR и NS для задачи с одним процессором для группы тестов 1 показаны в табл. 2.

В столбцах N_{IJR}, N_{JR} и N_{NS} табл. 2 приведено количество тестов в процентах, для которых были получены оптимальные решения алгоритмами IJR, JR и NS соответственно. В следующих трех столбцах содержится среднее значение отношения $R = C_{\max}/C_{\text{opt}}$ для этих алгоритмов по всем тестам.

Таблица 2. **Результаты работы приближенных алгоритмов для группы тестов 1**

n	N_{IJR}	N_{JR}	N_{NS}	R_{IJR}	R_{JR}	R_{NS}
50	81	5	74	1.00030	1.0097	1.00250
100	97	6	94	1.00006	1.0091	1.00010
300	96	0	88	1.00005	1.0015	1.00009
500	80	0	56	1.00004	1.0009	1.00020
1000	96	0	93	1.00000	1.00009	1.00002
2000	92	1	88	1.00001	1.0002	1.00002
5000	96	2	91	1.00000	1.0001	1.00001

Из табл. 2 видно, что по проценту оптимальных решений алгоритм IJR лучше алгоритмов NS и JR. Алгоритм JR очень редко получает оптимальное решение. Средняя относительная погрешность решения невелика у всех алгоритмов и составляет самое большее 0.005 % для алгоритма IJR, самое большее 0.97 % для алгоритма JR и 0.2 % для алгоритма NS.

В табл. 3 приведены результаты экспериментов, в которых проверялась зависимость работы алгоритмов от изменения константы K. Остальные столбцы этой таблицы аналогичны столбцам табл. 2. Из табл. 3 видно, что изменение константы K от 10 до 22 не оказывает заметного влияния на работу алгоритмов.

Таблица 3. **Зависимость работы алгоритмов от константы** K

n	K	N_{IJR}	N_{JR}	N_{NS}	R_{IJR}	R_{JR}	R_{NS}
100	10	98	2	88	1.00001	1.003	1.0001
100	14	95	0	91	1.00002	1.004	1.0005
100	15	92	1	89	1.00007	1.002	1.0010
100	16	93	2	91	1.00001	1.005	1.0001
100	18	97	0	87	1.00001	1.003	1.0001
100	20	92	4	91	1.00006	1.009	1.0001
100	22	93	2	88	1.00004	1.003	1.0002

Из теоретического анализа алгоритмов можно предположить, что наиболее трудные примеры будут иметь место, когда есть одно или два задания, значительно отличающиеся от остальных по длительности. Такие тесты генерировались в группах 2 и 3. В табл. 4 и 5 приведены результаты сравнения алгоритмов для тестов групп 2 и 3 соответственно. Для этих групп тестов рассматривался комбинированный алгоритм ICA, в котором из двух решений, полученных алгоритмами JR и IJR, выбиралось лучшее. В двух последних столбцах таблиц $R_{\mathrm{ICA}} = C_{\max}(S_A)/C_{\mathrm{opt}}$ — значение средней относительной погрешности для комбинированного алгоритма и N_{ICA} — количество оптимальных решений в процентах для комбинированного алгоритма.

Таблица 4. **Результаты сравнения алгоритмов для тестов группы 2**

n	K	N_{IJR}	N_{JR}	N_{NS}	R_{IJR}	R_{JR}	R_{NS}	R_{ICA}	N_{ICA}
100	10	51	23	25	1.02	1.05	1.04	1.005	58
100	14	29	47	48	1.06	1.03	1.03	1.004	62
100	15	25	48	49	1.05	1.04	1.04	1.007	59
100	16	53	21	34	1.01	1.04	1.01	1.004	69
100	18	46	46	49	1.05	1.04	1.03	1.005	71
100	20	24	15	16	1.05	1.06	1.03	1.007	33
100	22	44	29	33	1.02	1.03	1.03	1.006	57

Таблица 5. **Результаты сравнения алгоритмов для тестов группы 3**

n	K	N_{IJR}	N_{JR}	N_{NS}	R_{IJR}	R_{JR}	R_{NS}	R_{ICA}	N_{ICA}
100	10	48	35	42	1.04	1.05	1.04	1.006	59
100	14	55	29	29	1.02	1.04	1.04	1.005	68
100	15	25	43	44	1.06	1.03	1.02	1.006	57
100	16	36	39	41	1.05	1.04	1.04	1.005	66
100	18	40	40	41	1.06	1.05	1.05	1.004	72
100	20	42	25	26	1.08	1.01	1.01	1.005	58
100	22	24	14	16	1.06	1.07	1.07	1.008	36

Из табл. 4 и 5 видно, что для тестов с двумя или одной продолжительной работой процент оптимальных решений у алгоритмов IJR и NS уменьшается, а у алгоритма JR увеличивается. Для тестов с одной длительной работой (табл. 4) процент оптимальных решений у алгоритма IJR для трех наборов тестов примерно в 2 раза больше, чем у алгоритмов NS и JR. Для двух наборов приблизительно в 2 раза меньше. Для одного набора получено примерно одинаковое количество оптимальных решений. Для тестов с двумя продолжительными заданиями результаты аналогичные. Относительная погрешность решения увеличивается у всех алгоритмов и составляет в среднем от 1 до 6 %.

По результатам, приведенным в табл. 4 и 5, трудно выделить лучший алгоритм из JR, IJR и NS. Алгоритм NS строит два решения: одно — алгоритмом JR, а затем, используя полученную информацию, пытается его улучшить. Для тестов группы 1 это улучшение часто было значительным, а для групп 2 и 3 результаты алгоритма существенно не меняются. Значительно лучшими показателями обладает комбинированный алгоритм ICA: в нем сочетаются преимущества алгоритма JR, не допускающего невынужденных простоев, и предложенного автором алгоритма IJR, который их разрешает.

Применение комбинированного алгоритма ICA существенно улучшает относительную погрешность решения. Для этого алгоритма она составляет от 0.4 до 0.8 %. Худшие решения для алгоритма JR имели относительную погрешность 23 %, для алгоритма IJR — 19 %, а для комбинированного алгоритма ICA — только 7 %. В процессе тестирования не было получено теста, для которого алгоритмы JR и IJR построили решение с большой относительной погрешностью. Количество оптимальных решений также возрастает для комбинированного алгоритма ICA. Комбинированный алгоритм генерирует две перестановки, как и алгоритм NS, но его средняя относительная погрешность существенно меньше, а полученных оптимальных решений больше.

5. Заключение. В работе рассмотрена задача составления расписания для одного процессора с временами поступления и временами доставки заданий. Целью является минимизация общего времени выполнения всех заданий. Предложен новый алгоритм с гарантированной оценкой точности, равной $3/2$, и вычислительной сложностью $O(n \log n)$, в котором приоритет задания учитывается в первую очередь и допускаются простои процессора при выполнении определенных условий. Приведен пример, в котором достигается гарантированная оценка точности. Вычислительный эксперимент подтвердил практическую эффективность алгоритма.

Литература

1. *Graham R. L., Lawler E. L., Lenstra J. K., Rinnooy Kan A. H. G.* Optimization and approximation in deterministic sequencing and scheduling: A survey // Annals of Discrete Mathematics. 1979. Vol. 5. N 10. P. 287–326.

2. *Lenstra J. K., Rinnooy Kan A. H. G., Brucker P.* Complexity of machine scheduling problems // Annals of Discrete Mathematics. 1977. Vol. 1. P. 343–362.

3. *Лазарев А. А., Садыков Р. Р., Севастьянов С. В.* Схема приближенного решения задачи $1|r_i|L_{\max}$ // Дискретный анализ и исследование операций. Сер. 2. 2006. Т. 13. № 1. С. 57–76.

4. *Baker K. R.* Introduction to sequencing and scheduling. New York: John Wiley & Son, 1974. 318 p.

5. *Kanet J. J., Sridharan V.* Scheduling with inserted idle time: problem taxonomy and literature review // Operations Research. 2000. Vol. 48. N 1. P. 99–110.

6. *Григорьева Н. С.* Алгоритм ветвей и границ для задачи составления расписаний на параллельных процессорах // Вестник Санкт-Петербургского университета. Сер. 10. Прикладная математика. Информатика. Процессы управления. 2009. Вып. 1. С. 44–55.

7. *Григорьева Н. С.* Задача минимизации максимального временного смещения для параллельных процессоров // Вестник Санкт-Петербургского университета. Сер. 10. Прикладная математика. Информатика. Процессы управления. 2016. Вып. 4. С. 51–65.

8. *Grigoreva N. S.* Multiprocessor scheduling with inserted idle time to minimize the maximum lateness // Proceedings of the 7th Multidisciplinary International Conference of Scheduling: Theory and Applications. Prague: MISTA, 2015. P. 814–816.

9. *Grigoreva N. S.* Single machine inserted idle time scheduling with release times and due dates // Proceedings. DOOR2016. Vladivostoc, Russia. September 19–23, 2016. Ceur-WS. 2016. Vol. 1623. P. 336–343.

10. *Schrage L.* Optimal solutions to resource constrained network scheduling problems. Unpublished manuscript. 1971.

11. *Kise H., Ibaraki T., Mine H.* Performance analysis of six approximation algorithms for the one-machine maximum lateness scheduling problem with ready times // Journal of the Operations Research Society of Japan. 1979. N 22. P. 205–224.

12. *Potts C. N.* Analysis of a heuristic for one machine sequencing with release dates and delivery times // Operations Research. 1980. Vol. 28. P. 1436–1441.

13. *Hall L. A., Shmoys D. B.* Jackson's rule for single-machine scheduling: making a good heuristic better // Mathematics of Operations Research. 1992. Vol. 17. N 1. P. 22–35.

14. *Nowicki E., Smutnicki C.* An approximation algorithm for a single-machine scheduling problem with release times and delivery times // Discrete Applied Mathematics. 1994. Vol. 48. P. 69–79.

15. *Hall L. A., Shmoys D. B.* Approximation algorithms for constrained scheduling problems // Proceedings of the 30th IEEE Symposium on Foundations of Computer Science. IEEE Computer Society Press. 1989. P. 134–139.

16. *Mastrolilli M.* Efficient approximation schemes for scheduling problems with release dates and delivery times // Journal of Scheduling. 2003. Vol. 6. P. 521–531.

17. *Carlier J.* The one machine sequencing problem // European Journal of Operational Research. 1982. Vol. 11. P. 42–47.

Статья поступила в редакцию 12 февраля 2021 г.
Статья принята к печати 4 июня 2021 г.

Контактная информация:

Григорьева Наталья Сергеевна — канд. физ.-мат. наук, доц.; n.s.grig@gmail.com

3/2-approximation algorithm for a single machine scheduling problem

N. S. Grigoreva

St. Petersburg State University, 7–9, Universitetskaya nab., St. Petersburg, 199034, Russian Federation

For citation: Grigoreva N. S. 3/2-approximation algorithm for a single machine scheduling problem. *Vestnik of Saint Petersburg University. Applied Mathematics. Computer Science. Control Processes*, 2021, vol. 17, iss. 3, pp. 240–253. https://doi.org/10.21638/11701/spbu10.2021.302 (In Russian)

The problem of minimizing the maximum delivery times while scheduling tasks on a single processor is a classical combinatorial optimization problem. Each task u_i must be processed without interruption for $t(u_i)$ time units on the machine, which can process at most one task at time. Each task u_i has a release time $r(u_i)$, when the task is ready for processing, and a delivery time $q(u_i)$. Its delivery begins immediately after processing has been completed. The objective is to minimize the time, by which all jobs are delivered. In the Graham notation this problem is denoted by $1|r_j, q_j|C_{\max}$, it has many applications and it is NP-hard in a strong sense. The problem is useful in solving owshop and jobshop scheduling problems. The goal of this article is to propose a new 3/2-approximation algorithm, which runs in $O(n \log n)$ times for scheduling problem $1|r_j, q_j|C_{\max}$. An example is provided which shows that the bound of 3/2 is accurate. To compare the effectiveness of proposed algorithms, random generated problems of up to 5000 tasks were tested.

Keywords: single-machine scheduling problem, realize and delivery times, approximation algorithm, guarantee approximation ratio.

References

1. Graham R. L., Lawler E. L., Lenstra J. K., Rinnooy Kan A. H. G. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 1979, vol. 5, no. 10, pp. 287–326.

2. Lenstra J. K., Rinnooy Kan A. H. G., Brucker P. Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1977, vol. 1, pp. 343–362.

3. Lazarev A. A., Sadykov R. R., Sevastyanov S. V. Shema priblizhennogo resheniya zadachi $1|r_i|L_{\max}$ [A scheme of approximation solution of problem $1|r_i|L_{\max}$]. *Diskretny analiz i issedovanie operatcii* [*Discrete analyze and operation research*]. *Series 2*, 2006, vol. 13, no. 1, pp. 57–76. (In Russian)

4. Baker K. R. *Introduction to sequencing and scheduling.* New York, John Wiley & Son Publ., 1974, 318 p.

5. Kanet J. J., Sridharan V. Scheduling with inserted idle time: problem taxonomy and literature review. *Operations Research*, 2000, vol. 48, no. 1, pp. 99–110.

6. Grigoreva N. S. Algoritm vetvey i granits dlya zadachi sostavleniya raspisaniya na parallel'nykh processorah [Branch and bound algorithm for multiprocessor scheduling problem]. *Vestnik of Saint Petersburg University. Series 10. Applied Mathematics. Computer Science. Control Processes*, 2009, iss. 1, pp. 44–55. (In Russian)

7. Grigoreva N. S. Zadacha minimizatcii makcimalnogo vremennogo smeceniya dlya parallel'nykh processorov [Scheduling problem to minimize the maximum lateness for parallel processors]. *Vestnik of Saint Petersburg University. Series 10. Applied Mathematics. Computer Science. Control Processes*, 2016, iss. 4, pp. 51–65. (In Russian)

8. Grigoreva N. S. Multiprocessor scheduling with inserted idle time to minimize the maximum lateness. *Proceedings of the 7th Multidisciplinary International Conference of Scheduling. Theory and Applications.* Prague, MISTA Publ., 2015, pp. 814–816.

9. Grigoreva N. S. Single machine inserted idle time scheduling with release times and due dates. *Proceedings. DOOR2016.* Vladivostok, Russia, September 19–23, 2016. Ceur-WS, 2016, vol. 1623, pp. 336–343.

10. Schrage L. *Optimal solutions to resource constrained network scheduling problems.* Unpublished manuscript, 1971.

11. Kise H., Ibaraki T., Mine H. Performance analysis of six approximation algorithms for the one-machine maximum lateness scheduling problem with ready times. *Journal of the Operations Research Society of Japan*, 1979, no. 22, pp. 205–224.

12. Potts C. N. Analysis of a heuristic for one machine sequencing with release dates and delivery times. *Operations Research*, 1980, vol. 28, pp. 1436–1441.

13. Hall L. A., Shmoys D. B. Jackson's rule for single-machine scheduling: making a good heuristic better. *Mathematics of Operations Research*, 1992, vol. 17, no. 1, pp. 22–35.

14. Nowicki E., Smutnicki C. An approximation algorithm for a single-machine scheduling problem with release times and delivery times. *Discrete Applied Mathematics*, 1994, vol. 48, pp. 69–79.

15. Hall L. A., Shmoys D. B. Approximation algorithms for constrained scheduling problems. *Proceedings of the 30th IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society Press, 1989, pp. 134–139.

16. Mastrolilli M. Efficient approximation schemes for scheduling problems with release dates and delivery times. *Journal of Scheduling*, 2003, vol. 6, pp. 521–531.

17. Carlier J. The one machine sequencing problem. *European Journal of Operational Research*, 1982, vol. 11, pp. 42–47.

Received: February 12, 2021.
Accepted: June 04, 2021.

Author's information:

Natalia S. Grigoreva — PhD in Physics and Mathematics, Associate Professor; n.s.grig@gmail.com