ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ В ДИСКРЕТНОЙ МАТЕМАТИКЕ

УДК 519.8 DOI 10.17223/20710410/67/7

КОНСТРУКТИВНЫЕ АЛГОРИТМЫ ДЛЯ ЗАДАЧИ СОСТАВЛЕНИЯ РАСПИСАНИЙ НА ДВУХ ПРОЦЕССОРАХ С КРИТЕРИЕМ МАКСИМАЛЬНОГО ВРЕМЕННОГО СМЕЩЕНИЯ ПРИ УЧЁТЕ РАСПАРАЛЛЕЛИВАНИЯ И РАСХОДА ЭНЕРГИИ[1]

Ю. В. Захарова*, А. О. Евтина**

*Институт математики им. С. Л. Соболева СО РАН, г. Омск, Россия
**Омский государственный университет им. Ф. М. Достоевского, г. Омск, Россия

E-mail: yzakharova@ofim.oscsbras.ru, aevtina55@gmail.com

Проводится теоретический и экспериментальный анализ вычислительной сложности одной актуальной задачи теории расписаний, возникающей в компьютерных системах и приложениях. Особенностью постановки является возможность распараллеливания операций и учёт ресурсных ограничений, влияющих на длительности операций. Критерием выступает минимизация максимального временного смещения. Исследуется вопрос труднорешаемости задачи и предлагаются алгоритмы с гарантированными оценками точности. Результаты экспериментальных исследований показывают перспективность предложенных алгоритмов.

Ключевые слова: *расписание, ресурс, алгоритм, NP-трудность.*

CONSTRUCTIVE ALGORITHMS FOR THE SCHEDULING PROBLEM ON TWO PROCESSORS WITH THE MAXIMUM TIME OFFSET CRITERION TAKING INTO ACCOUNT PARALLELIZATION AND ENERGY CONSUMPTION

Y. V. Zakharova*, A. O. Evtina**

*Sobolev Institute of Mathematics, Omsk, Russia
**Dostoevsky Omsk State University, Omsk, Russia

We provide theoretical and experimental analysis of the computational complexity of one scheduling problem arising in computer systems and applications. The feature of the statement is the parallelization of operations and resource-dependent durations of operations. Here the processor speed affects the duration of operations and power consumption. For each operation the number of required processors is given. The criterion is the minimization of the maximum lateness under the given energy budget. We prove that the problem is NP-hard using the polynomial reduction from the well-known 3-Partition problem and constructing non-idle instances. An approximate algorithm with polynomial running time is proposed. The algorithm consists

[1]Работа выполнена в рамках госзадания ИМ СО РАН, проект FWNF-2022-0020.

of reducing the two-processor problem to the single-processor one. A convex program is proposed for solving the single-processor problem. We investigate the block properties of the solutions generated by the algorithm and show that their objective value is at most doubled optimum maximum lateness plus the maximal due date. The experimental results show the applicability of the proposed algorithm. We construct a series of instances in which the relative deviation from the lower bound is less than 15 %. We also experimentally identify a tendency in decreasing the relative deviation when the number of fully parallelizable operations increases. The theoretical analysis guarantees that the problem is polynomially solvable when all operations are fully parallelizable.

Keywords: *schedule, resource, algorithm, NP-hardness.*

Введение

В работе рассматриваются задачи составления расписания с малым числом процессоров, возникающие в компьютерных системах при планировании работы приложений на многоядерных процессорах. Современные компьютерные системы обуславливают необходимость или возможность обработки одной задачи несколькими процессорами, в микропроцессорных системах некоторые задачи должны выполняться на нескольких процессорах одновременно, кроме того, один процессор может использоваться для тестирования или проверки другого процессора [1].

Задачи составления расписаний широко исследованы при различных типах распараллеливания операций (задано число используемых процессоров или верхняя граница степени распараллеливания; заданы подмножества процессоров, которые могут использоваться для работы приложения). Исследованы вопросы труднорешаемости различных частных случаев, предложены модели целочисленного линейного программирования, алгоритмы с гарантированной точностью и метаэвристики [1].

Процессоры, выполняя различные приложения, имеют возможность совместного использования общих ресурсов (ядра, кэш-памяти, шины данных, энергии и т. д.), поэтому в задачах планирования могут быть заданы ограничения на ресурсы. Память относится к возобновимым ресурсам, при этом ограничивается возможность размещения операций на процессоре и совмещения с другими операциями в системах с общей памятью. Для вариантов задачи составления расписаний с возобновимыми ресурсами предложены алгоритмы динамического программирования, модели целочисленного программирования и метаэвристики [2–4]. Классу возобновимых ресурсов принадлежит также пропускная способность шины данных. Во время передачи информации по шине данных операции могут замедлять друг друга, когда выполняются одновременно, таким образом, скорость выполнения работы может меняться в зависимости от загрузки других ядер процессора [5]. Для задачи составления расписаний с учётом замедления операций при использовании шины данных построена модель целочисленного программирования, разработан алгоритм жадного типа и проведено экспериментальное исследование на реальных данных [6].

Энергия относится к невозобновимым ресурсам. Процессоры могут работать с различной скоростью, что влияет на длительность выполнения операций и потребление энергии. Задачам составления расписаний с учётом расхода энергии посвящены, например, работы [7–10]. Здесь рассматриваются задачи с бюджетными ограничениями на расход энергии при критериях, в основном, минимизации длины расписания и суммы моментов завершения операций, а также задачи с энергетическим критерием. Предлагаются приближённые алгоритмы с гарантированными оценками точности и

доказывается NP-трудность в частных случаях. Например, для задачи с распараллеливаемыми операциями на двух процессорах предложен 3/2-приближённый алгоритм для критерия «длина расписания» и 2-приближённый алгоритм для критерия «суммарное время завершения».

Задача с двумя устройствами актуальна, например, в вычислительных системах с конфигурацией из двух NUMA-узлов или с двухъядерными процессорами. Более того, подходы на основе декомпозиции могут использовать решение одно- и двухпроцессорных подзадач [1, 9].

В настоящей работе исследуется задача составления операций с учётом расхода энергии при возможности распараллеливания операций. В качестве критерия оптимизации в отличие от предыдущих исследований рассматривается минимизация максимального временного смещения $L_{\max}$. В п. 1 рассматривается задача, где учитывается только возможность распараллеливания операций, и предлагается приближённый алгоритм с гарантированной оценкой точности. В п. 2 осуществляется переход к задаче с учётом расхода энергии, доказывается её NP-трудность, адаптируется алгоритм из п. 1. Здесь же предлагается модель выпуклого программирования, позволяющая получить оптимальное решение за полиномиальное время в случае одного процессора. В п. 3 для оценки разработанных алгоритмов представлены результаты экспериментальных исследований.

1. Постановка задачи и алгоритм решения

Рассматривается задача составления расписания на одном или двух процессорах ($m = 1$ или 2) [11]. Для каждой операции (работы) j из $J = \{1, \ldots, n\}$ заданы длительность p_j, количество требуемых процессоров size_j и директивный срок d_j. Все операции поступают в момент времени 0, прерывания не допускаются. Процессоры обозначаются через M_1, M_2.

Рассматривается критерий минимизации максимального временного смещения $L_{\max} = \max\limits_{j \in J}\{L_j\}$, где $L_j = c_j - d_j$ — временное смещение операции j; c_j — момент окончания операции j, $j \in J$. Данная задача обозначается как $P|\mathrm{size}_j|L_{\max}$ [1] и является NP-трудной в сильном смысле [12], причём она остается NP-трудной, даже если все операции являются однопроцессорными [13].

В случае $m = 1$ и $\mathrm{size}_j = 1$ для всех операций $j \in J$ задача полиномиально разрешима. Для построения оптимального решения операции необходимо упорядочить по неубыванию их директивных сроков [13]. Обозначим через $\pi = (\pi_1, \ldots, \pi_n)$ указанную последовательность операций, тогда оптимальное временное смещение равно

$$L^*_{\max} = \max_{i \in \{1,\ldots,n\}} \left\{ \sum_{l=1}^{i} p_{\pi_l} - d_{\pi_i} \right\}.$$

1.1. Два процессора

Опишем приближённый алгоритм с гарантированной оценкой точности, основанный на сведении задачи с двумя процессорами к задаче с одним процессором.

Алгоритм A2-1-2 для задачи $P2|\mathrm{size}_j|L_{\max}$:

1) Задача P на двух процессорах сводится к задаче $P1$ на одном процессоре так, что директивные сроки не меняются, а длительности пересчитываются для каждой операции $j \in J$ по следующему правилу:
   - если $\mathrm{size}_j = 1$, то $p'_j = p_j/2$;
   - если $\mathrm{size}_j = 2$, то $p'_j = p_j$.

2) Строится последовательность операций π, соответствующая их упорядочению по неубыванию директивных сроков, то есть $d_{\pi_i} \leqslant d_{\pi_{i+1}}$. Эта последовательность даёт оптимальное решение задачи $P1$.
3) Для задачи P допустимое решение строится по следующему правилу: на шаге $i = 1, \ldots, n$ операция π_i назначается в расписание так, что если $\text{size}_{\pi_i} = 1$, то операция ставится на процессор с минимальным временем завершения текущего набора операций, а если $\text{size}_{\pi_i} = 2$, то операция ставится на два процессора с момента времени, когда они оба доступны.

Пример работы алгоритма представлен на рис. 1. Рассматривается пример задачи P с двумя процессорами и девятью работами; вектор длительностей: $p = (3, 2, 5, 4, 4, 2, 4, 7, 6)$, вектор директивных сроков: $d = (10, 3, 7, 6, 5, 8, 9, 13, 12)$, вектор размеров: $\text{size} = (2, 1, 2, 1, 1, 1, 1, 2, 1)$. При переходе к задаче $P1$ длительности однопроцессорных работ уменьшаются в 2 раза и новый вектор длительностей равен $p' = (3, 1, 5, 2, 2, 1, 2, 7, 3)$. Оптимальное расписание для задачи $P1$ соответствует перестановке $\pi = (2, 5, 4, 3, 6, 7, 1, 9, 8)$. Согласно этой же перестановке, завершаются операции в расписании для P.

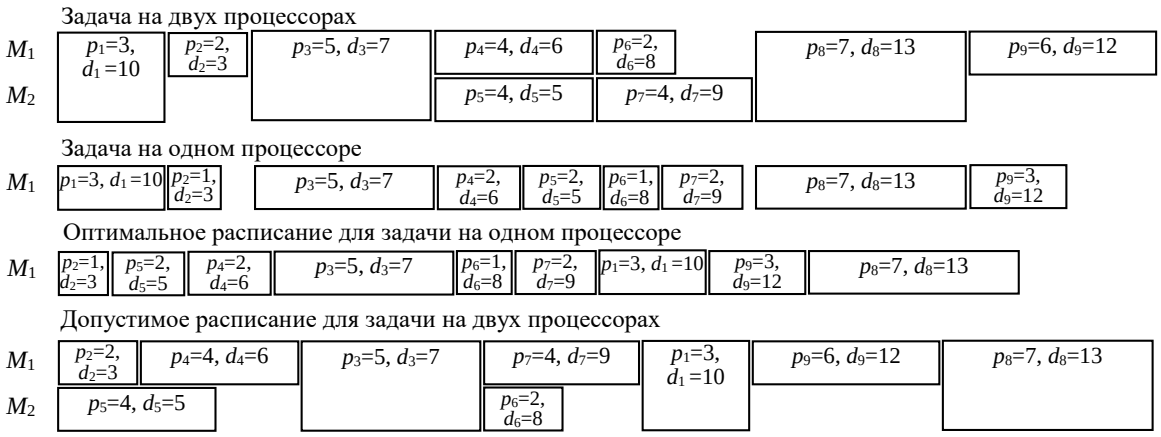

Рис. 1. Иллюстрация к работе алгоритма A2-1-2

Под ранним расписанием, соответствующим произвольной последовательности σ, будем понимать расписание, при котором операции завершают выполнение согласно σ, а начинаются в минимальные моменты времени, раньше которых они начаться не могут. Справедлива следующая

Лемма 1. Пусть последовательность σ задаёт порядок завершения операций; $L_{\max}(P, \sigma)$ — максимальное временное смещение в раннем расписании для задачи P на двух процессорах, соответствующем последовательности σ; $L_{\max}(P1, \sigma)$ — максимальное временное смещение в раннем расписании для задачи $P1$ на одном процессоре, соответствующем последовательности σ. Тогда $L_{\max}(P, \sigma) \geqslant L_{\max}(P1, \sigma)$.

Доказательство. Перестановка σ задает последовательность, согласно которой операции завершают выполнение. Обозначим через $c_j(P, \sigma)$ момент окончания операции $j \in J$ в задаче P на двух процессорах и через $c_j(P1, \sigma)$ — в задаче $P1$ на одном процессоре согласно раннему расписанию, соответствующему последовательности σ.

Для задачи P разделим последовательность на блоки: первый блок включает все однопроцессорные операции и первую встретившуюся двухпроцессорную операцию, во

второй блок входят все однопроцессорные операции до следующей двухпроцессорной операции и т. д. В каждом блоке при переходе к задаче на одном процессоре ставим на один процессор операции в том же порядке, в котором они заканчивались в двухпроцессорной задаче. Тогда для любой операции её время завершения на одном процессоре $c_j(P1,\sigma)$ станет меньше $c_j(P,\sigma)$ или останется таким же за счёт того, что длительности однопроцессорных заданий уменьшаются в 2 раза и предыдущие работы размещаются в том же порядке (так как предыдущие работы располагаются плотно и без простоев в двухпроцессорном расписании, в однопроцессорном расписании при уменьшении длительностей работ в 2 раза момент окончания работы может только уменьшиться). Длительности двухпроцессорных работ не меняются, поэтому моменты окончания тоже не увеличиваются. Тогда $c_j(P1,\sigma) \leqslant c_j(P,\sigma)$, $j \in J$. В итоге $L_{\max}(P,\sigma) = \max\limits_{j\in J}\{c_j(P,\sigma) - d_j\} \leqslant \max\limits_{j\in J}\{c_j(P1,\sigma) - d_j\} = L_{\max}(P1,\sigma)$. ■

Теорема 1. Если $L_{\max}(P)$ — значение целевой функции расписания, построенного для задачи P с помощью алгоритма A2-1-2, а $L^*_{\max}(P)$ — оптимальное значение целевой функции для задачи P, то $L_{\max}(P) \leqslant 2L^*_{\max}(P) + \max\limits_{j\in J} d_j$.

Доказательство. Момент окончания операции $j \in J$ в расписании, построенном алгоритмом A2-1-2 для задачи P, обозначим через $c_j(P)$, а в оптимальном расписании для задачи $P1$ — через $c_j(P1)$.

По построению в расписании, полученном для P алгоритмом A2-1-2, для операции $j \in J$ имеет место

$$L_j(P) = c_j(P) - d_j \leqslant 2c_j(P1) - d_j = c_j(P1) - d_j + c_j(P1) - d_j + d_j = 2(c_j(P1) - d_j) + d_j.$$

Отметим, что $c_j(P) \leqslant 2c_j(P1)$ для любой операции $j \in J$, это следует из того, что при переходе к двум процессорам момент окончания каждой операции увеличится не больше чем в 2 раза, так как длительности увеличиваются не больше чем в 2 раза. Тогда для любой операции j имеем $L_j(P) \leqslant 2(c_j(P1) - d_j) + d_j$, следовательно,

$$L_{\max}(P) \leqslant 2L^*_{\max}(P1) + \max_{j\in J} d_j,$$

где $L^*_{\max}(P1)$ есть оптимальное временное смещение для задачи $P1$.

Обозначим через σ_{opt} оптимальную последовательность завершения операций для P, тогда по лемме 1 имеем

$$L_{\max}(P1,\sigma_{\text{opt}}) \leqslant L_{\max}(P,\sigma_{\text{opt}}) = L^*_{\max}(P).$$

Ввиду того, что $L^*_{\max}(P1) \leqslant L_{\max}(P1,\sigma_{\text{opt}})$, получаем $L_{\max}(P) \leqslant 2L^*_{\max}(P) + \max\limits_{j\in J} d_j$. ■

2. Задача с учётом ресурсных ограничений

Перейдём к задаче составления расписания с ограничением на общее потребление невозобновимого ресурса — энергии. Для каждой операции $j \in J$ задан объём W_j, на основе которого вычисляются длительности p_j. Если процессор работает со скоростью s_j при выполнении операции j, то мгновенная мощность (потребление энергии в единицу времени) равна s_j^α, где $\alpha > 1$ — константа. Например, $\alpha = 1{,}11$ для Intel PXA270, $\alpha = 1{,}66$ для TCP offload engine, $\alpha = 3$ для CMOS устройств. Длительность операции j в таком случае равна $p_j = W_j/s_j = \left(W_j^\alpha/E_j\right)^{\alpha-1}$, а потребление энергии $E_j = (W_j/s_j)\, s_j^\alpha$. Общий доступный объём ресурса обозначим через E. Данная задача обозначается как $P|\text{size}_j, \text{energy}|L_{\max}$ в случае произвольного числа процессоров и $1|\text{energy}|L_{\max}$ — в случае одного процессора [1, 9].

Такого рода задачи с выпуклой функцией потребления ресурсов также возникают и в промышленном производстве [10]. Здесь длительности работ зависят от объёма потребляемого ресурса, а именно: длительность работы $j \in J$ равна $p_j(R_j) = (W_j/R_j)^\kappa$, где $\kappa \leqslant 1$ — константа; W_j — объём работы; R_j — объём потребляемого ресурса.

Случай $\kappa = 1$ соответствует многим реальным государственным и промышленным операциям, а случай $\kappa = 0{,}5$ — проектированию схем СБИС, где для отдельной задачи произведение площади кремния (ресурса) на квадрат затраченного времени равно константе (рабочей нагрузке).

2.1. Один процессор

Для задачи с одним процессором и заданным порядком следования операций $\pi = (\pi_1, \ldots, \pi_n)$ предлагается модель выпуклого программирования, которая разрешима за полиномиальное время с помощью метода эллипсоидов [14]:

$$L_{\max} \longrightarrow \min; \tag{1}$$

$$\sum_{i=1}^{j} p_{\pi_i} - d_{\pi_j} \leqslant L_{\max}, \quad j = 1, \ldots, n; \tag{2}$$

$$\sum_{j=1}^{n} W_j^\alpha \cdot p_{\pi_j}^{1-\alpha} = E; \tag{3}$$

$$L_{\max} \geqslant 0, \quad p_{\pi_j} \geqslant 0, \quad j = 1, \ldots, n. \tag{4}$$

Ограничение (2) соответствует вычислению временных смещений для операций, условие (3) ограничивает общее потребление ресурса.

Алгоритм A1E для задачи $1|\text{energy}|L_{\max}$:

1) Построить перестановку $\pi^{d\leqslant}$, соответствующую упорядочению операций по неубыванию директивных сроков.
2) Решить задачу (1)–(4) при перестановке $\pi^{d\leqslant}$.
3) Построить расписание по перестановке $\pi^{d\leqslant}$ с длительностями, найденными по модели (1)–(4).

Теорема 2. Алгоритм A1E позволяет найти оптимальное решение для задачи $1|\text{energy}|L_{\max}$ за полиномиальное время.

Доказательство. Чтобы решение задачи (1)–(4) было оптимальным для случая одного процессора, операции должны идти по неубыванию их директивных сроков согласно перестановке $\pi^{d\leqslant}$. Допустим, что это не так. Поменяем местами две любые подряд идущие операции, будем считать, что операции имеют номера i и $i+1$ и директивные сроки $d_i < d_{i+1}$. Их моменты окончания изменятся: для операции i временное смещение станет равным $c_i + p_{i+1} - d_i = c_{i+1} - d_i$, а для операции $i+1$ оно будет равно $c_{i+1} - p_i - d_{i+1}$. Таким образом, $c_{i+1} - d_i > c_i - d_i$, $c_{i+1} - d_i > c_{i+1} - d_{i+1}$ и $c_{i+1} - p_i - d_{i+1} < c_{i+1} - d_{i+1}$. Значит, после перестановки максимум из двух временных смещений достигается на $c_{i+1} - d_i$, и эта величина больше, чем временные смещения до обмена местами. ■

2.2. Два процессора

Докажем NP-трудность задачи на двух процессорах с распараллеливанием операций в случае длительностей работ, зависящих от потребления ресурсов, и опишем полиномиальный приближённый алгоритм с гарантированной оценкой точности.

Теорема 3. Задача $P2|\text{size}_j, \text{energy}|L_{\max}$ является NP-трудной в сильном смысле.

Доказательство. Используем подход, предложенный в [7]. Рассмотрим классическую задачу 3-Разбиение [15], где имеется $3k$ элементов с суммарным весом $\sum\limits_{i=1}^{3k} e_i = kB$. Вопрос заключается в том, существует ли разбиение на k подмножеств $A_1, \ldots, A_k$, где $\sum\limits_{i \in A_1} e_i = \ldots = \sum\limits_{i \in A_k} e_i = B$. В случае положительного ответа каждое подмножество содержит 3 элемента.

Построим пример задачи $P2|\text{size}_j, \text{energy}|L_{\max}$. Пусть число работ $n = 6k$, а объёмы, число используемых процессоров и директивные сроки определены следующим образом:

$$W_j = e_j,\ d_j = 4kB,\ \text{size}_j = 1,\ j = 1, \ldots, 3k,$$
$$W_j = 2B,\ d_j = (4(j-3k)-2)B,\ \text{size}_j = 1,\ j = 3k+1, \ldots, 4k,$$
$$W_j = 3B,\ d_j = (4(j-4k)-1)B,\ \text{size}_j = 1,\ j = 4k+1, \ldots, 5k,$$
$$W_j = B,\ d_j = 4(j-5k)B,\ \text{size}_j = 2,\ j = 5k+1, \ldots, 6k.$$

Общий объём энергии положим равным $E = 8kB$. Требуется ответить на вопрос, существует ли расписание с $L_{\max} \leqslant 0$, то есть когда все работы выполняются до их директивных сроков.

Заметим, что суммарный объем всех работ равен $8kB$, а максимальный директивный срок равен $4kB$ при наличии двух процессоров. Значит, в любом допустимом расписании нет простоев, следовательно, работы могут выполняться только с единичной скоростью (ввиду выпуклости функции расчёта мгновенной мощности). В таком случае допустимое расписание имеет структуру, представленную на рис. 2, и существует тогда и только тогда, когда в задаче 3-Разбиение ответ «да». ■

Рис. 2. Иллюстрация к доказательству NP-трудности (структура допустимого расписания)

Для построения приближённого решения используется подход из п. 1.1 и алгоритм для однопроцессорной задачи из п. 2.1: сначала реализуется переход к задаче на одном процессоре путём уменьшения объёмов, далее решается задача выпуклого программирования и осуществляется возврат к задаче на двух процессорах путём увеличения длительностей.

Алгоритм A2-1-2E для задачи $P2|\text{size}_j, \text{energy}|L_{\max}$:

1) Задача PE на двух процессорах сводится к задаче $PE1$ на одном процессоре так, что директивные сроки не меняются, а объёмы пересчитываются для каждой операции $j \in J$ по следующему правилу:
   - если $\text{size}_j = 1$, то $W'_j = W_j/2$;
   - если $\text{size}_j = 2$, то $W'_j = W_j$.

   Объём энергии уменьшается в 2 раза: $E' = E/2$.
2) Строится оптимальное решение для задачи $PE1$ на одном процессоре с помощью алгоритма A1E, где операции упорядочиваются по неубыванию директивных сроков согласно перестановке $\pi^{d\leqslant}$.

3) Вычисляются длительности операций для задачи PE:
   — если $\text{size}_j = 1$, то $p_j = 2p'_j$;
   — если $\text{size}_j = 2$, то $p_j = p'_j$.
4) Для задачи PE допустимое решение строится по следующему правилу: на шаге $i = 1, \ldots, n$ операция $\pi_i^{d\leqslant}$ назначается в расписание так, что если $\text{size}_{\pi_i^{d\leqslant}} = 1$, то операция ставится на процессор с минимальным временем завершения текущего набора операций, а если $\text{size}_{\pi_i^{d\leqslant}} = 2$, то операция ставится на два процессора с момента времени, когда они оба доступны.

Теорема 4. Если $L_{\max}(PE)$ — значение целевой функции расписания, построенного для задачи PE алгоритмом A2-1-2E, а $L^*_{\max}(PE)$ — оптимальное значение целевой функции для задачи PE, то $L_{\max}(PE) \leqslant 2L^*_{\max}(PE) + \max\limits_{j\in J} d_j$.

Доказательство. Для алгоритма A2-1-2E сохраняются оценки точности, доказанные в теореме 1 для алгоритма A2-1-2 и задачи без учёта расхода энергии. Здесь принимается во внимание тот факт, что алгоритм A1E для однопроцессорной задачи позволяет получить её оптимальное решение за полиномиальное время, а оптимальное временное смещение для задачи PE1 является нижней оценкой на целевую функцию для задачи с двумя процессорами. Нижняя оценка гарантируется за счёт того, что уменьшение объёмов операций и общего бюджета энергии в 2 раза и переход к однопроцессорной задаче равносильны релаксации исходной задачи до варианта, когда все работы могут распараллеливаться на два процессора с уменьшением длительностей в 2 раза. ■

3. Вычислительный эксперимент

Для экспериментального исследования алгоритмов A2-1-2 и A2-1-2E были случайным образом сгенерированы примеры с равномерным распределением. Для задачи построено 25 серий. В каждой серии генерируется 50 задач с 50 операциями с заданными параметрами. Операции рассматривались двух типов — малой (из интервала $[10, 20]$) и большой (из интервала $[100, 120]$) длительности. Тип операции задаётся так, что с вероятностью P_{small} операция имеет малую длительность, а с вероятностью $(1 - P_{\text{small}})$ — большую. Число используемых процессоров для каждой операции задаётся по правилу: один процессор выбирается с вероятностью P_{proc} и два процессора — с вероятностью $(1 - P_{\text{proc}})$. Вероятности выбираются из множества $\{0,\ 0{,}25,\ 0{,}5,\ 0{,}75,\ 1\}$. Директивные сроки задаются в зависимости от общего объёма операций: для группы из первых 25 серий директивные сроки задаются от 25 до 50 % объёма (суммы всех длительностей работ, деленной на два), для второй группы из следующих 25 серий — от 45 до 90 % от объёма.

Для анализа результатов эксперимента вычисляются среднее относительное отклонение от нижней границы и стандартное отклонение. В табл. 1 представлены значения среднего относительного отклонения от нижней границы для первой группы (с округлением до сотых).

Значения среднего относительного отклонения увеличиваются с ростом доли операций малой длительности. Среднее относительное отклонение уменьшается с увеличением доли двухпроцессорных операций. Отметим, что значения не превышают 15 %.

Поскольку значения целевой функции зависят от правила генерации директивных сроков, целесообразно провести исследование для серий задач с большими, относительно общего объёма операций, директивными сроками. В табл. 2 представлены значения

Таблица 1

Среднее относительное отклонение в % результата алгоритма A2-1-2 от нижней границы для первой группы примеров (в скобках указано стандартное отклонение)

Доля операций малой длительности	Доля двухпроцессорных операций					Среднее
	0	0,25	0,5	0,75	1	
0	1,34 (1,21)	6,67 (3,30)	5,50 (2,17)	2,98 (2,42)	0	3,30 (1,82)
0,25	2,75 (1,23)	7,36 (3,36)	6,54 (2,15)	5,27 (2,12)	0	4,38 (1,77)
0,5	3,71 (2,17)	8,37 (2,56)	6,90 (3,87)	6,74 (4,12)	0	5,14 (2,54)
0,75	4,78 (3,37)	9,54 (4,79)	8,18 (4,33)	7,05 (4,25)	0	5,91 (3,35)
1	1,98 (1,24)	7,93 (2,35)	8,18 (2,45)	4,66 (1,98)	0	4,50 (1,60)
Среднее	2,91 (1,46)	7,97 (2,89)	7,02 (2,66)	5,34 (2,66)	0	4,65 (2,21)

среднего относительного отклонения от нижней границы для второй группы (с округлением до сотых).

Таблица 2

Среднее относительное отклонение в % результата алгоритма A2-1-2 от нижней границы для второй группы примеров (в скобках указано стандартное отклонение)

Доля операций малой длительности	Доля двухпроцессорных операций					Среднее
	0	0,25	0,5	0,75	1	
0	8,27 (4,29)	34,95 (5,42)	23,67 (3,49)	15,35 (9,37)	0	16,44 (4,51)
0,25	13,82 (6,43)	31,37 (4,25)	29,36 (3,42)	24,21 (8,47)	0	19,75 (4,51)
0,50	17,11 (8,15)	38,22 (8,37)	34,44 (7,99)	26,20 (4,35)	0	23,19 (5,77)
0,75	32,07 (7,41)	42,56 (5,22)	46,17 (6,24)	35,63 (8,39)	0	31,28 (5,45)
1	11,68 (5,37)	37,47 (7,28)	34,08 (10,11)	23,35 (9,87)	0	21,31 (6,52)
Среднее	16,59 (6,11)	36,91 (6,25)	33,54 (6,25)	24,94 (8,09)	0	22,40 (5,35)

Заметим, что для второй группы значения среднего относительного отклонения при генерации двухпроцессорных операций с долями 0,25 и 0,5 отличаются несущественно, кроме случая, когда все операции большой длительности. В остальном сохраняются тенденции, наблюдаемые для первой группы; значения не превышают 55 %.

По результатам эксперимента для обеих групп можно заметить, что среднее относительное отклонение увеличивается с увеличением директивных сроков. С уменьшением доли двухпроцессорных операций увеличивается среднее относительное отклонение от нижней границы, за исключением крайних случаев: когда все операции двухпроцессорные (нижняя граница даёт оптимальное решение) и когда все операции однопроцессорные, что даёт меньшее значение среднего относительного отклонения, чем случаи с наличием двухпроцессорных операций.

Как видно из результатов вычислительного эксперимента, на примерах, сгенерированных случайным образом с равномерным распределением, среднее относительное отклонение от нижней границы существенно меньше, чем получено для алгоритма в худшем случае. Отметим необходимость дальнейшего теоретического и экспериментального исследования отклонения нижней границы от оптимального решения для различных классов примеров.

Результаты для алгоритма A2-1-2E идентичны и в обобщённом виде представлены в табл. 3. Для решения выпуклой программы, соответствующей подзадаче с од-

ним процессором, использовался коммерческий пакет Baron (сервер NEOS — `https://neos-server.org/neos/`). Общий объём энергии полагался равным $E = 1000$.

Таблица 3

Среднее относительное отклонение в % результата алгоритма A2-1-2E от нижней границы (в скобках указано стандартное отклонение)

Доля операций	0	0,25	0,5	0,75	1
	Первая группа примеров				
Двухпроцессорных	5,17 (2,3)	8,25 (3,5)	7,13 (3,3)	6,18 (3,2)	0
Малой длительности	5,23 (1,3)	5,27 (1,1)	7,43 (3,3)	7,45 (2,1)	5,32 (2,2)
	Вторая группа примеров				
Двухпроцессорных	18,25 (5,1)	36,35 (7,3)	33,11 (5,1)	24,35 (6,1)	0,09
Малой длительности	19,42 (6,4)	22,25 (7,3)	25,45 (8,4)	32,40 (8,3)	21,13 (7,2)

Заключение

Проведено исследование задачи составления расписаний на двух процессорах с возможностью распараллеливания и учётом ограничения на потребление энергии. Доказана NP-трудность и предложен полиномиальный приближённый алгоритм с гарантированной оценкой точности в худшем случае. Для случая одного процессора предложена модель выпуклого программирования, позволяющая получать оптимальное решение за полиномиальное время. Результаты экспериментального исследования показали перспективность используемого подхода. В дальнейшем представляет интерес анализ генерической сложности рассматриваемых задач и обобщение результатов на случай произвольного числа процессоров, например на основе декомпозиции.

ЛИТЕРАТУРА

1. *Drozdowski M.* Scheduling for Parallel Processing. Dordrecht: Springer, 2009. 386 p.
2. *Сервах В. В.* Эффективно-разрешимый случай задачи календарного планирования с возобновимыми ресурсами // Дискретн. анализ и исслед. опер. 2000. Сер. 2. Т. 7. № 1. С. 75–82.
3. *Коваленко Ю. В.* О задаче календарного планирования с возобновимым ресурсом // Автомат. и телемех. 2012. № 6. С. 140–153.
4. *Goncharov E. N.* An improved genetic algorithm for the resource-constrained project scheduling problem // N. Olenev, Yu. Evtushenko, M. Jaćimović, et al. (eds). Advances in Optimization and Applications. Springer, Cham, 2022. P. 35–47.
5. *Zhuravlev S., Blagodurov S., and Fedorova A.* Addressing shared resource contention in multicore processors via scheduling // Comput. Archit. News. 2010. V. 38. No. 1. P. 129–142.
6. *Еремеев А. В., Сахно М. Ю.* Построение расписания для многоядерного процессора с учетом взаимного влияния работ // Выч. мет. программирование. 2023. Т. 24. № 1. С. 115–126.
7. *Kononov A. and Kovalenko Y.* On speed scaling scheduling of parallel jobs with preemption // LNCS. 2016. V. 9869. P. 309–321.
8. *Kononov A. and Zakharova Y.* Speed scaling scheduling of multiprocessor jobs with energy constraint and total completion time criterion // Intern. J. Artif. Intell. 2023. V. 21. No. 2. P. 109–129.
9. *Gerards M. E. T., Hurink J. L., and Holzenspies P. K. F.* A survey of offline algorithms for energy minimization under deadline constraints // J. Scheduling. 2016. V. 19. P. 3–19.
10. *Shabtay D. and Kaspi M.* Parallel machine scheduling with a convex resource consumption function // Europ. J. Oper. Res. 2006. V. 173. No. 1. P. 92–107.

11. https://www2.informatik.uni-osnabrueck.de/knust/class/ — Complexity Results for Scheduling Problems. 2024.
12. *Lee C.-Y. and Cai X.* Scheduling one and two-processor tasks on two parallel processors // IIE Trans. 1999. V. 31. P. 445–455.
13. *Lenstra J. K., Rinnooy Kan A. H. G., and Brucker P.* Complexity of machine scheduling problems // Ann. Discrete Math. 1977. V. 1. P. 343–362.
14. *Grotschel M., Lovasz L., and Schrijver A.* Geometric Algorithms and Combinatorial Optimizations. Heidelberg: Springer, 2009. 332 p.
15. *Garey M. and Johnson D.* Computers and Intractability. A Guide to the Theory of NP-completeness. N.Y.: W. H. Freeman and Company, 1979. 338 p.

REFERENCES

1. *Drozdowski M.* Scheduling for Parallel Processing. Dordrecht, Springer, 2009. 386 p.
2. *Servakh V. V.* Effektivno-razreshimyy sluchay zadachi kalendarnogo planirovaniya s vozobnovimymi resursami [An efficiently solvable case of the scheduling problem with renewable resources]. Diskretn. Analiz i Issled. Oper., 2000, ser. 2, vol. 7, no. 1, pp. 75–82. (in Russian)
3. *Kovalenko Yu. V.* On the calendar planning problem with renewable resource. Autom. Remote Control, 2012, vol. 73, no. 6, pp. 1046–1055.
4. *Goncharov E. N.* An improved genetic algorithm for the resource-constrained project scheduling problem. N. Olenev, Yu. Evtushenko, M. Jaćimović, et al. (eds). Advances in Optimization and Applications, Springer, Cham, 2022, pp. 35–47.
5. *Zhuravlev S., Blagodurov S., and Fedorova A.* Addressing shared resource contention in multicore processors via scheduling. Comput. Archit. News, 2010, vol. 38, no. 1, pp. 129–142.
6. *Eremeev A. V. and Sakhno M. Yu.* Postroenie raspisaniya dlya mnogoyadernogo protsessora s uchetom vzaimnogo vliyaniya rabot [Multi-core processor scheduling with respect to the mutual influence of jobs]. Vych. Met. Programmirovanie, 2023, vol. 24, no. 1, pp. 115–126. (in Russian)
7. *Kononov A. and Kovalenko Y.* On speed scaling scheduling of parallel jobs with preemption. LNCS, 2016, vol. 9869, pp. 309–321.
8. *Kononov A. and Zakharova Y.* Speed scaling scheduling of multiprocessor jobs with energy constraint and total completion time criterion. Intern. J. Artif. Intell., 2023, vol. 21, no. 2, pp. 109–129.
9. *Gerards M. E. T., Hurink J. L., and Holzenspies P. K. F.* A survey of offline algorithms for energy minimization under deadline constraints. J. Scheduling, 2016, vol. 19, pp. 3–19.
10. *Shabtay D. and Kaspi M.* Parallel machine scheduling with a convex resource consumption function. Europ. J. Oper. Res., 2006, vol. 173, no. 1, pp. 92–107.
11. https://www2.informatik.uni-osnabrueck.de/knust/class/ — Complexity Results for Scheduling Problems, 2024.
12. *Lee C.-Y. and Cai X.* Scheduling one and two-processor tasks on two parallel processors. IIE Trans., 1999, vol. 31, pp. 445–455.
13. *Lenstra J. K., Rinnooy Kan A. H. G., and Brucker P.* Complexity of machine scheduling problems. Ann. Discrete Math., 1977, vol. 1, pp. 343–362.
14. *Grotschel M., Lovasz L., and Schrijver A.* Geometric Algorithms and Combinatorial Optimizations. Heidelberg, Springer, 2009. 332 p.
15. *Garey M. and Johnson D.* Computers and Intractability. A Guide to the Theory of NP-completeness. N.Y., W. H. Freeman and Company, 1979. 338 p.