МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»(НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Механико-математический факультет

Кафедра теоретической кибернетики

Направление подготовки 02.03.01 — Математика и компьютерные науки

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА МАГИСТРА

Храмовой Антонины Павловны

Новый алгоритм решения двухмашинной задачи open shop и его приложение к одной задаче маршрутизации

«К защите допущена»
Заведующий кафедрой,
д.ф.-м.н., проф.
Ерзин А. И./ _____
(подпись, МП)
«____» _____ 2020 г.

Научный руководитель
к.ф.-м.н.
с.н.с ИМ СО РАН
Черных И. Д./ _____
(подпись, МП)
«____» _____ 2020 г.

Дата защиты: «____»_____2020г.

Новосибирск, 2020

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ ОБОЗНАЧЕНИЙ **3**

ВВЕДЕНИЕ **5**

1. Предварительные сведения **9**

2. Алгоритмы для задачи $O2||C_{\max}$ **11**

2.1. Алгоритм Гонзалеза-Сани 11

2.2. Алгоритм Пинедо-Шраге 12

2.3. Алгоритм де Верра 13

2.4. Алгоритм Сопера 15

2.5. Новый алгоритм 16

3. Задача Open Shop с маршрутизацией и нефиксированной базой **18**

3.1. Алгоритм для задачи на цикле и его доказательство 18

3.2. Следствия теоремы 21

ЗАКЛЮЧЕНИЕ **23**

ЛИТЕРАТУРА **25**

ПЕРЕЧЕНЬ ОБОЗНАЧЕНИЙ

Обозн.	**Расшифровка**
$\mathcal{J}$	Множество всех работ в примере задачи.
n	Мощность множества $\mathcal{J}$, если явно не указано иного.
J_j	j-ая работа, $J_j \in \mathcal{J}$.
M_i	i-ая машина, $i = 1, 2$.
a_j, b_j	Операции работы J_j на M_1 и M_2 соответственно; также длины этих операций.
$x \to y$	Операция y может быть выполнена только после операции x.
ℓ_i	Нагрузка машины M_i. $\ell_{\max} = \max \ell_i$.
d_j	Длина работы J_j, $d_j = a_j + b_j$. $d_{\max} = \max d_j$.
$C_{\max}(S)$	Длина расписания S для задачи open shop.
$\overline{C}$	Стандартная нижняя оценка на длину расписания для задачи open shop; $\overline{C} = \max\{\ell_{\max}, d_{\max}\}$.
T_i, T_i^*	Обход и длина оптимального обхода транспортной сети машиной M_i.
T^*	$T^* = T_1^*$ в случае, если $T_1^* = T_2^*$.
$R_{\max}(S)$	Длина расписания S для задачи open shop с маршрутизацией.
$\overline{R}$	Нижняя оценка на длину расписания для задачи open shop с маршрутизацией и нефиксированной базой; $\overline{R} = \max\{\ell_{\max} + T_i^*, d_{\max}\}$.
$S(\pi)$	Раннее расписание, строящееся по упорядоченному списку работ π так, как описано в параграфе 2.5.

Следующие обозначения используются в трёхместной записи задач теории расписаний ...

Обозн.	**Расшифровка**
Om	... для m-машинной задачи open shop (на I месте).
ROm	... для m-машинной задачи open shop с маршрутизацией.
$G = X$	... для задач с транспортной сетью G структуры X (II).
$variable - depot$	... для задач с нефиксированной базой.
Qtt	... для задач с однородными временами перемещения.
Rtt	... для задач с независимыми временами перемещения.
$easy - TSP$	... для задач с транспортной сетью, на которой задача коммивояжёра разрешима за полиномиальное время.
$C_{\max}$	Длина расписания для задачи без маршрутизации (III).
$R_{\max}$	Длина расписания для задачи с маршрутизацией.

ВВЕДЕНИЕ

Задача *open shop* была впервые рассмотрена в работе [1] и является одной из классических в области теории расписаний. В рамках задачи требуется построить расписание выполнения множества работ несколькими машинами. Каждая машина должна выполнить единственную операцию каждой работы, причём длительность выполнения любой операции известна заранее. Операции одной работы могут быть выполнены в произвольном порядке, в отличие от похожей известной задачи flow shop, в которой каждая следующая машина не может приступить к некоторой работе раньше, чем предыдущая машина закончит её выполнять. Ни одна машина не может выполнять более одной операции за раз, также как и ни одна работа не может выполняться на двух машинах одновременно. Цель задачи — минимизировать *длину расписания*, равную $C_{\max}$, величине, соответсвующей времени завершения последней операции. В традиционной трёхместной системе обозначения задач теории расписаний [2] задача open shop на m машинах обозначается $Om||C_{\max}$.

Известно, что задача $O3||C_{\max}$ NP-трудна [1]. Однако оптимальное решение чуть более простой задачи $O2||C_{\max}$ может быть найдено за линейное время. Существует несколько алгоритмов для поиска этого решения, начиная с уже упомянутой работы Гонзалеза и Сани [1] и продолжая алгоритмами Пинедо и Шраге [3], де Верра [4] и Сопера [5]. Эти алгоритмы и свойства соответствующих расписаний описываются в главе 2 этой работы.

Одним из результатов этой работы является новый алгоритм для задачи $O2||C_{\max}$. Хотя результирующее расписание по своим свойствам похоже на расписание, получаемое алгоритмом Сопера, новый алгоритм обладает несколькими преимуществами. Во-первых, его описание и доказательство оптимальности значительно проще по сравнению с предложенными Сопером, а

во-вторых, этот алгоритм возможно применить для решения более общей задачи — *задачи open shop с маршрутизацией.*

Задача open shop с маршрутизацией впервые представлена в работах [6, 7] и может быть описана следующим образом. Каждая работа расположена в некоторой вершине *транспортной сети*, задаваемой неориентированным рёберно-взвешенным графом G. Вес ребра — это время, затрачиваемое машиной на перемещение между двумя соответствующими вершинами. Чтобы выполнить работу, машина должна переместиться по рёбрам транспортной сети в вершину, где эта работа находится. Считается, что по одному ребру одновременно может передвигаться неограниченное количество машин. Все машины начинают работу в одной вершине — *базе* — и должны вернуться в эту вершину по завершению выполнения всех операций. Цель задачи — минимизировать длину расписания $R_{\max}$, величину, соответствующую моменту завершения последнего действия некоторой машины, будь то перемещение в базу либо выполнение операции, расположенной в базе. Задача на m машинах обозначается $ROm||R_{\max}$. Если в постановке требуется указать структуру X (дерево, цикл и т. п.) транспортной сети G, то используется запись $ROm|G = X|R_{\max}$.

Таким образом, задача open shop с маршрутизацией является обобщением как классической задачи open shop (если все рёбра траспортной сети нулевого веса), так и метрической задачи коммивояжёра (если все операции нулевой длины), и поэтому задача в общем случае NP-трудна в сильном смысле. Более того, даже "простейший" подслучай этой задачи $RO2|G = K_2|R_{\max}$ тоже является NP-трудным, как было доказано в первой работе по open shop с маршрутизацией [6]. Другой важный результат в этой связи — FPTAS для этого "простейшего" случая [8], который частично основывается на структурных

свойствах оптимального расписания, получаемого по алгоритму Гонзалеза-Сани.

В работе [9] предложены два новых направления исследования постановки с маршрутизацией:

1. Кроме *фиксированной* базы, то есть заданной по условию, можно также рассматривать *нефиксированную* базу, то есть определяемую в результате работы алгоритма. Во втором случае используется запись $ROm|variable-depot|R_{\max}$.

2. Время перемещений разных машин по одному и тому же ребру не обязательно должно *совпадать*. В частности, можно рассмотреть *однородные* времена перемещений, то есть, для любых двух машин M_{i_1} и M_{i_2} существует скаляр $k > 0$ такой, что по любому ребру время перемещения машиной M_{i_1} в k раз дольше, чем то же время пемещения машиной M_{i_2}. Также можно говорить о *независимых* временах перемещения. В трёхместной записи задачи используются обозначения Qtt или Rtt соответственно.

Очевидно, что в общем случае задача $RO2|variable - depot|R_{\max}$ NP-трудна в сильном смысле, поскольку содержит задачу коммивояжёра в качестве подслучая. В работе [9] также доказано, что задача $RO2|Rtt, G = tree, variable - depot|R_{\max}$ полиномиально разрешима. Однако до сих пор ничего не было известно о сложности задачи $RO2|variable - depot|R_{\max}$, или её Qtt- и Rtt-обобщений, в случае, когда соответствующая задача коммивояжёра может быть решена за полиномиальное время в силу особенностей структуры транспортной сети G или её матрицы расстояний. Мы используем обозначение $easy - TSP$ для такой ситуации, по аналогии с [10], где был приведён $\frac{4}{3}$-приближённый алгоритм для задачи $RO2|easy - TSP|R_{\max}$.

В работе предложен оптимальный линейный алгоритм для задачи $RO2|Rtt, G = cycle, variable - depot|R_{\max}$, из которого можно вывести новый линейный алгоритм для классической задачи open shop, о котором было сказано ранее. Важным следствием этого результата является полиномиальная разрешимость двух задач: $RO2|Qtt, easy - TSP, variable - depot|R_{\max}$, что отвечает на поставленный выше вопрос о сложности задачи маршрутизации с условием $easy - TSP$, и $RO2|Rtt, G = cactus, variable - depot|R_{\max}$, что покрывает упомянутый результат из [9]. Также приводится приближённый результат для задачи $RO2|Qtt, variable - depot|R_{\max}$.

Работа организована следующим образом. В главе 1 приведены некоторые понятия и обозначения, необходимые для последующего изложения. Глава 2 представляет собой обзор всех известных алгоритмов для задачи $O2||C_{\max}$ со сравнением структурных свойств соответствующих расписаний, который завершается новым алгоритмом. Глава 3 содержит основной результат работы: описание алгоритма для задачи open shop с маршрутизацией и независимыми временами перемещения, его доказательство и следствия о полиномиальной разрешимости связанных задач.

1. Предварительные сведения

Ниже приведены несколько определений и обозначений, широко распространённых в теории расписаний и активно использующихся в последующих главах этой работы.

Через $\mathcal{J} = \{J_j | j = 1, \ldots, n\}$ обозначается множество из n работ. Все рассматриваемые в работе задачи сформулированы в системе из двух машин M_1 и M_2.

Обозначим за a_j, b_j две операции работы $J_j \in \mathcal{J}$, выполняющиеся соответственно машинами M_1 и M_2. Иногда с помощью a_j и b_j мы будем обозначать не только сами операции, но и их длины.

Нагрузка машины M_i — это сумма длин всех операций, назначенных на эту машину, обозначается за ℓ_i. При этом $\ell_{\max} = \max\limits_{i=1,2} \ell_i$. Также определим *длину* j-ой работы $d_j = a_j + b_j$ и величину $d_{\max} = \max\limits_{J_j \in \mathcal{J}} d_j$.

Для задачи open shop определим её *стандартную нижнюю оценку*: $\overline{C} = \max\{\ell_{\max}, d_{\max}\}$. Никакое допустимое расписание задачи open shop не может быть короче, чем $\overline{C}$, а достижение расписанием этой оценки доказывает его оптимальность.

Многие алгоритмы для двухмашинной задачи open shop предполагают построение *раннего расписания*. Это расписание, для которого задан порядок выполнения операций на каждой машине и каждой работе, и в котором каждая операция начинается в наиболее ранний момент, допускаемый этим порядком. Несложно заметить, что такое расписание можно построить за линейное время.

Многие обозначения, например, уточняющие особенности постановки той или иной задачи теории расписаний, используются в ходе работы в соответствии с их определением во введении. Некоторые понятия и обозначения

вводятся по ходу изложения в местах, обусловленных контекстом. Вниманию читателя также предлагается перечень, приведённый в начале этой работы и наиболее полно и последовательно отражающий все принятые в работе обозначения.

2. Алгоритмы для задачи $O2||C_{\max}$

2.1. Алгоритм Гонзалеза-Сани

Первый алгоритм для задачи $O2||C_{\max}$ был предложен Т. Гонзалезом и С. Сани в работе [1] — в той же, в которой задача была впервые рассмотрена. Основная идея решения в том, чтобы разделить множество работ на два подмножества и по определённому правилу выбрать одну работу, порядок выполнения операций в которой будет отличаться от прочих. В результате составления раннего расписания с учётом этой работы будет получено допустимое и оптимальное решение.

Описание алгоритма дано в том виде, как оно приведено в [11].

Алгоритм $\mathcal{A}^{GS}$:

1. Разделить $\mathcal{J}$ на два подмножества $\mathcal{J}_1 = \{J_j | a_j \leq b_j\}$ и $\mathcal{J}_2 = \mathcal{J} \setminus \mathcal{J}_1$.

2. Выбрать две работы J_p и J_q такие, что $a_p = \max\limits_{J_j \in \mathcal{J}_1} a_j$ и $b_q = \max\limits_{J_j \in \mathcal{J}_2} b_j$.

3. Если $a_p < b_q$, то пусть S — раннее расписание с порядком выполнения операций $\{a_j | J_j \in \mathcal{J}_1 \setminus \{J_p\}\} \to \{a_j | J_j \in \mathcal{J}_2\} \to a_p$ на машине M_1 и $b_p \to \{b_j | J_j \in \mathcal{J}_1 \setminus \{J_p\}\} \to \{b_j | J_j \in \mathcal{J}_2\}$ на машине M_2, причём операции из одного подмножества выполняются в произвольном порядке, хотя и совпадающим на двух машинах. Для любой работы, кроме J_p, порядок выполнения операций $a_j \to b_j$.

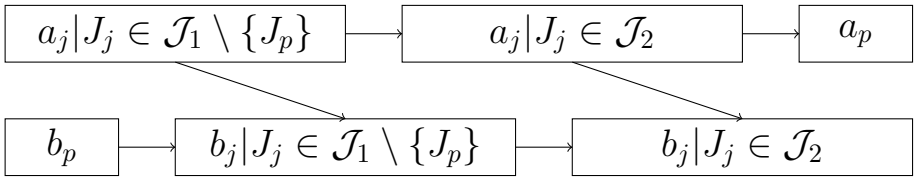

Рис. 1. Схема раннего расписания S в случае $a_p < b_q$.

Иначе, пусть S — раннее расписание с порядком выполнения операций

$a_q \to \{a_j | J_j \in \mathcal{J}_2 \setminus \{J_q\}\} \to \{a_j | J_j \in \mathcal{J}_1\}$ для машины M_1 и $\{b_j | J_j \in \mathcal{J}_2 \setminus \{J_q\}\} \to \{b_j | J_j \in \mathcal{J}_1\} \to b_q$ для машины M_2. Для любой работы, кроме J_q, порядок выполнения операций $b_j \to a_j$.

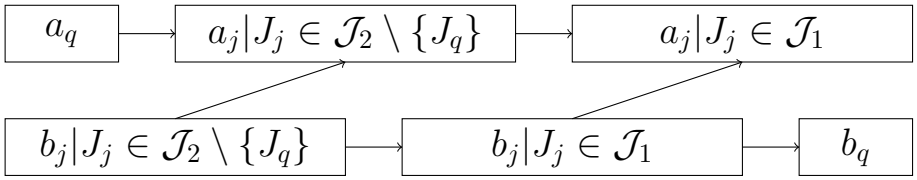

Рис. 2. Схема раннего расписания S в случае $a_p \geq b_q$.

4. **Output** S.

Работа J_p в первом случае и работа J_q во втором случае называется *диагональной*, за расположение её операций в указанном раннем расписании. Заметим, что в случае $a_p < b_q$, в силу выбора подмножеств, либо машина M_1 простаивает перед выполнением операции a_p, либо машина M_2 простаивает во время выполнения блока операций работ из множества $\mathcal{J}_2$; никаких других простоев в расписании нет. В первом случае, длина расписания $C_{\max}(S) = \max\{\ell_2, d_p\}$, а во втором имеем $C_{\max}(S) = \ell_1$. Так или иначе, $C_{\max}(S) = \overline{C}$, а значит, решение оптимально. Аналогичные рассуждения верны и для случая $a_p \geq b_q$.

2.2. Алгоритм Пинедо-Шраге

Этот алгоритм был представлен в 1982 г. математиками Пинедо и Шраге. Если в алгоритме Гонзалеза-Сани мы использовали ранние расписания, то здесь строится *плотное* расписание. В нём для машины M_i задан упорядоченный список $L_i = (j_1, j_2, \ldots, j_n)$ неповторяющихся индексов, каждый из которых соответствует операции некоторой работы J_{j_k}, $k = 1, \ldots, n$. В любой момент времени t, в который расписание на отрезке $[0, t]$ уже построено, а

машина M_i простаивает, на неё назначается первая доступная операция из списка L_i.

Алгоритм $\mathcal{A}^{PS}$ [3, теорема 2.1.1]:

1. Выбрать работу J_k с самой длинной операцией. Не умаляя общности считаем, что это операция b_k.

2. Построить плотное расписание S такое, что список L_1 начинается с k, а список L_2 заканчивается индексом k — то есть, операция b_k откладывается настолько долго, насколько это возможно. Все остальные операции могут располагаться в списке в любом порядке.

3. **Output** S.

В таком расписании только одна операция может выполняться после простоя: ведь если машина M_i простаивает до того, как выполнена операция работы J_j, то другая машина должна была выполнять свою операцию работы J_j во всё время простоя на машине M_i; иначе работа J_j была бы доступной раньше, что привело бы к противоречию с плотностью расписания. Доказательство оптимальности решения легко следует из этого его свойства и выбора операции b_k.

2.3. Алгоритм де Верра

В алгоритме де Верра [4], вместо того, чтобы определять порядок выполнения для каждой операции, множество всех работ $\mathcal{J}$ разбивается на три непересекающихся подмножества, и порядок выполнения определяется для этих самых подмножеств. То есть, в отличие от алгоритма Гонзалеза-Сани, в котором тоже происходит разделение всех работ на три подмножества, в

алгоритме де Верра допускается разный порядок выполнения операций на двух машинах в рамках одного подмножества.

Следующее описание алгоритма де Верра может быть найдено в [12].

Алгоритм $\mathcal{A}^W$:

1. Найти индекс $s \geq 2$ некоторой работы такой что $\sum_{j=1}^{s-1} d_j \leq \ell_{\max}$ и $\sum_{j=1}^{s} d_j > \ell_{\max}$.

2. Определить три подмножества работ: $\mathcal{J}_1 = \{J_1, J_2, \ldots, J_{s-1}\}$, $\mathcal{J}_2 = \{J_s\}$, and $\mathcal{J}_3 = \{J_{s+1}, \ldots, J_n\}$.

   Несложно убедиться в том, что $\sum_{J_j \in \mathcal{J}_k} d_j \leq \ell_{\max}$, $k = 1, 2, 3$.

3. При необходимости перенумеровать подмножества так что $\sum_{J_j \in \mathcal{J}_1} a_j \geq \sum_{J_j \in \mathcal{J}_2} b_j$ и $\sum_{J_j \in \mathcal{J}_1} b_j \geq \sum_{J_j \in \mathcal{J}_3} a_j$.

4. Построить раннее расписание S такое, что у машины M_1 порядок выполнения подмножеств $\mathcal{J}_1 \to \mathcal{J}_2 \to \mathcal{J}_3$, а у машины M_2 этот порядок $\mathcal{J}_2 \to \mathcal{J}_3 \to \mathcal{J}_1$. Порядок выполнения операций внутри каждого подмножества произвольный. Для любой работы $J_j \notin \mathcal{J}_1$ порядок выполнения операций соответствует $a_j \to b_j$, а для любой другой работы порядок $b_j \to a_j$.

5. **Output** S.

Как и в алгоритме $\mathcal{A}^{GS}$, из построения подмножеств следует, что либо машина M_1 простаивает во время выполнения операций из подмножества $\mathcal{J}_3$, и тогда машина M_2 вообще не простаивает и $C_{\max}(S) = \ell_2$, либо машина M_2 простаивает во время выполнения операций из подмножества $\mathcal{J}_1$, откуда следует $C_{\max}(S) = \max\{d_{\max}, \ell_1\}$. В обоих случаях $C_{\max}(S) = \overline{C}$, что и требуется.

2.4. Алгоритм Сопера

В расписании Сопера сохраняется идея выбора диагональной работы, но её роль в построении этого расписания в некоторым смысле обратная. Если в алгоритме $\mathcal{A}^{GS}$ сначала выбиралась диагональная работа, а потом вокруг неё строилось всё остальное расписание, то алгоритм Сопера производит циклический поиск по всем расписаниям типа flow shop для $n-1$ из n работ, где единственная нерассматриваемая работа меняется на каждой итерации. Как только в результате такого поиска будет найдено "хорошее" flow shop расписание, недостающая работа добавляется в него в качестве диагональной.

Далее в этом параграфе считаем $J_j = J_{j \mod n}$. За S'_t обозначим перестановочное расписание для двухмашинной задачи flow shop на множестве работ $\mathcal{J} \setminus \{J_{t-1}\}$ с порядком работ $(J_t, J_{t+1}, \ldots, J_n, J_1, \ldots, J_{t-2})$. Индекс $k(t)$ соответствует *критической работе* в расписании S'_t, то есть, такой работе, что $C_{\max}(S'_t) = \sum\limits_{j=t}^{k(t)} a_j + \sum\limits_{j=k(t)}^{t-2} b_j$.

Алгоритм $\mathcal{A}^S$ [5]:

1. Полагая $t = 1$, вычислить $C_{\max}(S'_t)$.

2. **while** $C_{\max}(S'_t) > \max\{\ell_1, \ell_2\}$ **or** $t \le k(1)$ **do**

   (a) Переопределить $t \leftarrow t + 1$.

   (b) Вычислить $C_{\max}(S'_t) = C_{\max}(S'_{t-1}) - a_{t-1} + b_{t-2}$, поскольку критический путь остаётся таким же.

3. Построить расписание S для задачи $O2||C_{\max}$ по расписанию S'_t, выполняя операцию b_{t-1} до начала выполнения всех операций в S'_t на машине M_2, а операцию a_{t-1} — после завершения выполнения всех операций в S'_t на M_1.

4. **Output** S.

Корректность алгоритма $\mathcal{A}^S$ и оптимальность соответствующего расписания доказывается рядом технических выкладок и вытекает из нескольких очевидных неравенств, основывающихся на устройстве просматриваемых перестановочных flow shop расписаний.

2.5. Новый алгоритм

Для списка работ $\pi = (J_1, J_2, \dots J_n)$ определим раннее расписание $S(\pi)$ такое, что:

(a) у машины M_1 порядок выполнения операций $a_2 \to a_3 \cdots \to a_n \to a_1$;

(b) у машины M_2 порядок выполнения операций $b_1 \to b_2 \to b_3 \cdots \to b_n$;

(c) для любой работы кроме J_1 порядок выполнения операций $a_j \to b_j$.

Обозначение π^{+k} используется для смещённого списка
$(J_k, J_{k+1}, \dots, J_n, J_1, \dots, J_{k-1})$.

Рассмотрим следующий алгоритм для задачи $O2||C_{\max}$.

Алгоритм $\mathcal{A}$:

1. При необходимости перенумеровать машины таким образом, чтобы выполнялось $\ell_1 \leq \ell_2$.

2. Пусть $\pi = (J_1, J_2, \dots J_n)$ — произвольный список работ. Построить расписание $S(\pi)$.

3. Если $C_{\max}(S(\pi)) = \overline{C}$, то **Output** $S(\pi)$.

   Иначе,

   (a) Пусть J_k — работа, которая выполняется после последнего построя на машине M_2 в расписании $S(\pi)$.

(*b*) **Output** $S(\pi^{+k})$.

Как в подходе Гонзалеза и Сани, ключевой элемент в построении расписания — выбор диагональной работы. Однако в отличие от алгоритма $\mathcal{A}^{GS}$, в котором задаётся конкретное правило выбора этой работы, не зависящее от дополнительных построений, здесь диагональная работа сначала выбирается произвольно, а затем выбирается на основе построенного расписания. Структура результирующего расписания совпадает со структурой расписания, получающегося в результате работы алгоритма $\mathcal{A}^{S}$, но мы строим расписание иначе: здесь не требуется ни построение flow shop расписаний, ни циклический поиск по множеству расписаний.

Теорема 1. Алгоритм $\mathcal{A}$ *возвращает расписание длины* $\overline{C}$ *за линейное время* $O(n)$.

Доказательство этой теоремы напрямую следует из доказательства теоремы 2, изложенного в следующей главе.

3. Задача Open Shop с маршрутизацией и нефиксированной базой

3.1. Алгоритм для задачи на цикле и его доказательство

Напомним, что времена перемещений в задаче open shop с маршрутизацией называются однородными, если веса рёбер транспортной сети отличаются для разных машин домножением на скаляр. Пусть G — циклическая транспортная сеть для двухмашинной задачи open shop с маршрутизацией и однородными временами перемещения. За T_i обозначим оптимальный обход этой транспортной сети машиной M_i, $i \in \{1, 2\}$, а за T_i^* — длину этого обхода.

Для задачи open shop с маршрутизацией с нефиксированной базой определим нижнюю оценку $\overline{R} = \max\{\ell_i + T_i^*, d_{\max}\}$. Разумеется, если времена перемещений двух машин совпадают, то $T_1^* = T_2^*$, и эту величину мы будем обозначать за T^*.

Рассмотрим следующее обобщение Алгоритма $\mathcal{A}$ для задачи $RO2|Rtt, G = cycle, variable - depot|R_{\max}$.

Алгоритм $\mathcal{A}'$:

1. Пусть $\pi = (J_1, J_2, \ldots J_n)$ — список работ в порядке их появления при обходе цикла G. Полагаем, что работа J_1 расположена в вершине v. Выбрать вершину v в качестве базы.

2. При необходимости перенумеровать машины таким образом, чтобы выполнялось $\ell_1 + T_1^* \leq \ell_2 + T_2^*$.

3. Построить расписание $S(\pi)$.

4. Если $R_{\max}(S(\pi)) = \overline{R}$, то **Output** $S(\pi)$.

   Иначе,

(a) Пусть работа J_k, расположенная в вершине u, выполняется после последнего построя на второй машине в расписании $S(\pi)$.

(b) Выбирая u в качестве базы, **Output** $S(\pi^{+k})$.

Теорема 2. Алгоритм $\mathcal{A}'$ *возвращает расписание длины* $\overline{R}$ *за линейное время* $O(n)$.

Доказательство. Заметим, что если $R_{\max}(S(\pi)) > \overline{R}$, то машина M_2 простаивает. Действительно, если не простаивает M_2, то должна простаивать M_1. По определению $S(\pi)$, машина M_1 может простаивать только до начала выполнения операции a_1, что возможно только если операция b_1 выполняется во время простоя. Тогда $R_{\max}(S_\pi) = \max\{d_1, \ell_2 + T_2^*\} \leq \overline{R}$, противоречие.

Пусть t — момент времени, в который заканчивается последний простой на машине M_2, который также является моментом начала выполнения операции b_k для некоторого индекса $k \in \{1, \ldots, n\}$. Рассмотрим расписание $S'(\pi)$, полученное сдвигом операций $b_1, \ldots, b_{k-1}$ в расписании $S(\pi)$ вправо так, что машина M_2 простаивает только до выполнения операции a_2, как показано на рис. 3. Длина расписания остаётся той же. Определим *блоки* (упорядоченные множества операций и времён перемещений между соответствующими вершинами) A_1, A_2, B_1, и B_2 следующим образом:

$A_1 = \boxed{\to a_2 \to \cdots \to a_k}$; $A_2 = \boxed{\to a_{k+1} \to \cdots \to a_n \to a_1}$;

$B_1 = \boxed{b_1 \to \cdots \to b_{k-1} \to}$; $B_2 = \boxed{b_k \to \cdots \to b_n \to}$.

Стрелки обозначают соответствующие времена перемещений.

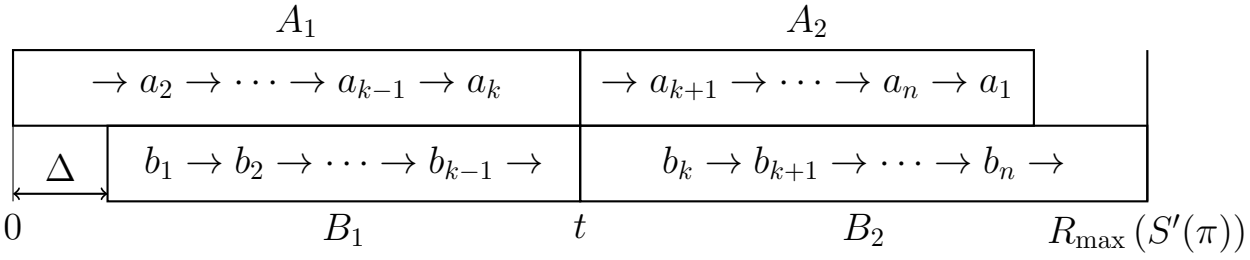

Рис. 3. Схема расписания $S'(\pi)$

Пусть Δ — момент, в который начинается выполнение блока B_1, так что $R_{\max}(S'(\pi)) = \Delta + \ell_2 + T_2^*$. Отметим, что выполнение блока A_2 заканчивается в момент $\ell_1 + T_1^*$, и $\ell_1 + T_1^* \leq \ell_2 + T_2^*$ влечёт $\Delta \leq R_{\max}(S'(\pi)) - (\ell_1 + T_1^*)$.

Рассмотрим расписание, полученное перестановкой блоков A_2 с A_1 и B_2 с B_1, как показано на рис. 4.

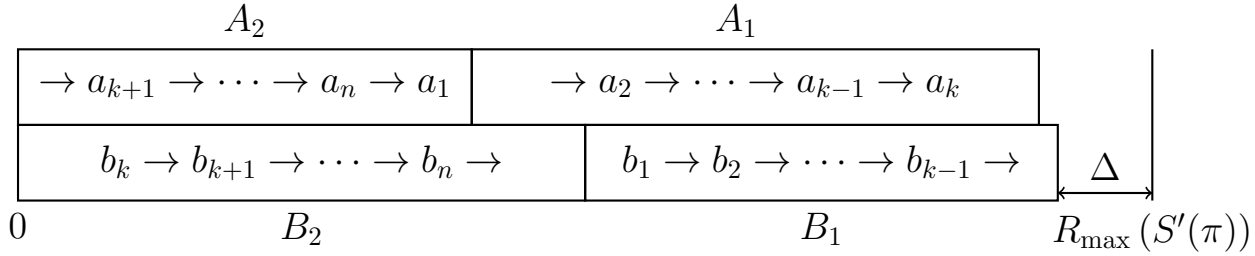

Рис. 4. Результат перестановки блоков

Допустимость этого расписания следует из указанного выше неравенства $\Delta \leq R_{\max}(S'(\pi)) - (\ell_1 + T_1^*)$, если только операции работы J_k не накладываются друг на друга, и на самом деле, это в точности расписание $S(\pi^{+k})$. Если всё же операция b_k заканчивается позже, чем начинается операция a_k, то мы получаем расписание $S(\pi^{+k})$ сдвигом операции a_k вправо соответствующим образом. По построению расписания, машина M_2 не простаивает, а M_1 может простаивать только перед выполнением a_k, так что $R_{\max}(S(\pi^{+k}))$ равна либо длине работы J_k, либо $R_{\max}(S(\pi^{+k})) = \max\{\ell_1 + T_1^*, \ell_2 + T_2^*\} \leq \overline{R}$. Отсюда следует $R_{\max}(S(\pi^{+k})) = \overline{R}$, что и требовалось.

Поскольку раннее расписание может быть построено за линейное время, алгоритм $\mathcal{A}'$ также возвращает решение за линейное время. □

Отметим, что задача $O2||C_{\max}$ является частным случаем задачи $RO2|variable-depot|R_{\max}$, если положить, что все времена перемещений равны нулю. Значит, алгоритм $\mathcal{A}$, изложенный в предыдущей главе, напрямую следует из только что описанного алгоритма $\mathcal{A}'$.

3.2. Следствия теоремы

Основной принцип работы алгоритма $\mathcal{A}'$ состоит в построении раннего расписания такого, что порядки выполнения операций на двух машинах совпадают с точностью до циклической перестановки работ, и обе машины следуют одному и тому же оптимальному обходу транспортной сети. В связи с этим далее мы рассмотрим два подслучая задачи $RO2|Rtt, variable - depot|R_{\max}$, полиномиальную разрешимость которых легко доказать с использованием алгоритма $\mathcal{A}'$.

Нетрудно заметить, что если времена перемещений однородны, то любой оптимальный обход первой машиной M_1 также является оптимальным обходом и для второй машины M_2. Поэтому, если дана транспортная сеть G с условием $easy - TSP$, то простая модификация алгоритма $\mathcal{A}'$ возвращает решение за полиномиальное время.

Следствие 3.1. *Задача $RO2|Qtt, easy - TSP, variable - depot|R_{\max}$ разрешима за полиномиальное время $O(n + t_{TSP})$, где t_{TSP} — время, необходимое для решения задачи коммивояжёра на траспортной сети G.*

В случае, если для задачи коммивояжёра имеется приближённое решение, мы можем использовать алгоритм $\mathcal{A}'$, чтобы получить приближённое решение такой же степени для задачи $RO2|Qtt, variable-depot|R_{\max}$. В частности, применяя алгоритм Кристофидеса-Сердюкова [13, 14], мы получаем следующее

Следствие 3.2. *Для задачи $RO2|Qtt, variable - depot|R_{max}$ существует $\frac{3}{2}$-приближённый алгоритм.*

Алгоритм $\mathcal{A}'$ можно применить только если оптимальные обходы для машин M_1 и M_2 совпадают. Однако в общем случае это неверно: именно, когда

времена перемещений независимы и структура графа G не уточняется. В этом следствии, структура графа G обеспечивает совпадение тех самых оптимальных обходов. *Кактусом* назовём граф, у которого все блоки — циклы либо рёбра.

Следствие 3.3. *Задача* $RO2|Rtt, G = cactus, variable - depot|R_{\max}$ *разрешима за линейное время* $O(n)$.

Наконец, если времена перемещений независимы, имеет место следующий результат.

Теорема 3. *Для задачи* $RO2|Rtt, variable - depot|R_{\max}$, *пусть* τ — *обход транспортной сети* G, *найденный за полиномиальное время, такой что длина этого обхода* $\tau^* = \rho_i \cdot T_i^*$ *для некоторого* ρ_i, $i \in \{`1, 2\}$.

Тогда для задачи существует алгоритм A *такой, что*

$$R_{max}(S_A) \leq \max_{i=1,2}\{\ell_i + \rho_i \cdot T_i^*, d_{max}\} \leq \max(\rho_1, \rho_2) \cdot R_{max}^*.$$

ЗАКЛЮЧЕНИЕ

Задачи теории расписаний с маршрутизацией часто возникают в приложениях, поскольку, в отличие от классических моделей, учитывают задержки при выполнении операций различными машинами. Однако описанная в работе постановка — это лишь один из способов учитывать такие задержки. Например, в [15] предлагается перемещать не машины, а работы между неподвижными машинами, что на самом деле очень похоже на рассматриваемую нами постановку, поскольку в open shop всегда можно менять местами работы и машины. Другой известный подход [16] предполагает объединение работ в группы и задания машинам времени, которое они тратят на переключение между этими группами. В терминах нашей постановки это соответствует расположению группы работ в одной вершине транспортной сети, а вес любого ребра соответствует времени переключения между группами. Однако важным отличием этих подходов от рассматриваемой постановки является отсутствие базы.

В представленной работе был предложен линейный алгоритм для задачи $RO2|Rtt, G = cycle, variable - depot|R_{\max}$ и показана полиномиальная разрешимость для задач $RO2|Rtt, G = cactus, variable-depot|R_{\max}$ и $RO2|Qtt, easy-TSP, variable-depot|R_{\max}$. Последний результат интересен в связи с тем, что, согласно нему, задача open shop с маршрутизацией и нефиксированной базой сложна лишь постольку, поскольку сложна задача коммивояжёра на соответствующей транспортной сети. Это отличает задачу от её вариации с фиксированной базой, где NP-трудность доказана даже для траспортной сети $G = K_2$, на которой задача коммивояжёра решается, конечно же, полиномиально. То же справедливо и для описанного выше подхода [16], в котором база отсутствует вовсе, но для простейшего случая из двух машин и двух групп работ

задача также NP-трудна. Таким образом, результат демонстрирует важность базы и её свойств в постановке задачи.

Как было упомянуто во введении, для простейшего случая задачи open shop с маршрутизацией $RO2|G = K_2|R_{\max}$ существует FPTAS, который частично основан на структурных свойствах оптимального расписания для классической задачи open shop, получаемого по алгоритму Гонзалеза-Сани. Было бы интересно посмотреть, могут ли структурные свойства расписания, полученного алгоритмом $\mathcal{A}$, быть использованы в рамках этого исследования.

Одним из ключевых свойств алгоритма $\mathcal{A}$ является то, что диагональная работа не определяется заранее, но выбирается в процессе построения расписания. В то же время, несложно понять, что в оптимальном расписании $S(\pi)$ для некоторого списка работ π не всякая работа может выполнять роль диагональной. Это поднимает естественный вопрос о том, какими свойствами может определяться работа, которая может быть диагональной, и как можно было бы её найти независимо от соответствующего ей расписания, или возможно ли даже построить оптимальное расписание вокруг работы, которая заранее была определена как потенциально диагональная.

ЛИТЕРАТУРА

[1] Gonzalez T. F., Sahni S. Open shop scheduling to minimize finish time // J. ACM. — 1976. — Vol. 23, no. 4. — P. 665–679.

[2] Chapter 9. Sequencing and scheduling: Algorithms and complexity / Eugene L. Lawler, Jan Karel Lenstra, Alexander H.G Rinnooy Kan, David B. Shmoys // Logistics of Production and Inventory. — Elsevier, 1993. — Vol. 4 of Handbooks in Operations Research and Management Science. — P. 445–522.

[3] Pinedo M., Schrage L. Stochastic shop scheduling: a survey // Deterministic and stochastic scheduling / Ed. by M. A.H Dempster, Jan Karl Lenstra, Alexander H.G Rinnooy Kan. — Springer, Dordrecht, 1982. — Vol. 84 of NATO Advanced Study Institute Series. — P. 181–196.

[4] de Werra D. Graph-theoretical models for preemptive scheduling // Advances in Project Scheduling. — 1989. — P. 171–185. — Access mode: http://infoscience.epfl.ch/record/88562.

[5] Soper A. J. A cyclical search for the two machine flow shop and open shop to minimise finishing time // Journal of Scheduling. — 2013. — nov. — Vol. 18, no. 3. — P. 311–314.

[6] Averbakh I., Berman O., Chernykh I. A 6/5-approximation algorithm for the two-machine routing open-shop problem on a two-node network // European Journal of Operational Research. — 2005. — Vol. 166, no. 1. — P. 3–24.

[7] Averbakh I., Berman O., Chernykh I. The routing open-shop problem on a network: complexity and approximation // European Journal of Operational Research. — 2006. — Vol. 173, no. 2. — P. 531–539.

[8] Кононов А. В. О цеховой задаче открытого типа на двух машинах с маршрутизацией в двухвершинной сети // Дискретный анализ и исследование операций. — 2012. — Т. 19, № 2. — С. 54–74.

[9] Chernykh I. Routing open shop with unrelated travel times // Discrete Optimization and Operations Research — 9th International Conference, DOOR 2016, Vladivostok, Russia, September 19-23, 2016, Proceedings. — 2016. — P. 272–283.

[10] Chernykh I., Kononov A. V., Sevastyanov S. Efficient approximation algorithms for the routing open shop problem // Computers & OR. — 2013. — Vol. 40, no. 3. — P. 841–847.

[11] Brucker P. Scheduling algorithms. — Springer-Verlag Berlin Heidelberg, 2007. — ISBN: 978-3-540-69515-8.

[12] Esswein C., Billaut J.-C., Strusevich V. A. Two-machine shop scheduling: compromise between flexibility and makespan value // European Journal of Operational Research. — 2005. — dec. — Vol. 167, no. 3. — P. 796–809.

[13] Christofides N. Worst-case analysis of a new heuristic for the travelling salesman problem. — Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburg, PA, 1976.

[14] Сердюков А. О некоторых экстремальных обходах в графах // Управляемые системы. Сб. науч. тр. — 1978. — Т. 17. — С. 76–79.

[15] Complexity results for flow-shop and open-shop scheduling problems with transportation delays / Peter Brucker, Sigrid Knust, T. C. Edwin Cheng, Natalia Shakhlevich // Annals of Operations Research. — 2004. — no. 129. — P. 81–106.

[16] Kleinau U. Two-machine shop scheduling problems with batch processing // Mathematical and Computer Modelling. — 1993. — mar. — Vol. 17, no. 6. — P. 55–66.