Задачи теории расписаний

Григорьева Наталья Сергеевна

СПбГУ

2023

План доклада

- Что такое задача составления расписания?
- Методы решения задач составления расписания.
- Основные модели задач теории расписаний
- Алгоритмыпостроения оптимальных и допустимых расписаний.
- Задача составления расписания для одного процессора.

Введение

Задачи упорядочения носят самый общий характер. Они возникают там, где существует возможность выбора той или иной очередности выполнения работ: при распределении работ на производстве, составлении расписания приземления самолетов, составлении расписания движения поездов, обслуживании клиентов в обслуживающих системах, управления проектами, управлении персоналом и т.д. Результаты ТР применяются при создании аппаратно-программных комплексов параллельной обработки данных и при усовершенствовании микроархитектуры процессоров.

Описание системы машин (процессоров)

Постановка задачи в теории расписаний начинается с описания системы машин (процессоров) и множества работ.
Различают параллельные системы машин и системы различных машин.
Для параллельной системы предполагается, чсто каждая работа может выполняться любой машиной. Машины могут быть идентичными или иметь разную производительность. Параллельные машины могут быть объединены в сеть.
Для системы конвейерного типа имеются различные машины и для каждой работы известно, на какой машине она выполняется. Работа может і состоять из нескольких операций, каждая из которых выполняется на своей машине.

Множество работ (заданий)

Для каждой задачи задается набор следующих величин:

1. - длительность выполнения работы $t(i)$
2. - момент поступления работы $r(i)$.
3. - плановый (директивный) срок $d(i)$. Эта величина представляет собой момент, к которому i-я работа должна быть выполнена.
4. - вес работы $w(i)$.

На множестве работ может быть задано отношение частичного порядка. Могут быть разрешены прерывания работ.

Основные определения

Будем говорить, что составить расписание значит найти для каждой работы (операции) i время начала выполнения этой работы (операции) τ_i и указать процессор, на котором она должна выполняться.

Если допустимы прерывания, то необходимо задать время начала каждого интервала выполнения задания, суммарная длина всех интервалов, в которых выполняется задание i, должна быть равна t_i.

В каждый момент времени процессор может выполнять только одно задание.

Расписание называется допустимым, если выполняются все наложенные ограничения.

Критерии эффективности расписаний

Рассматриваются различные критерии эффективности расписаний: один из самых распрастраненных критериев — длина расписания

$$C_{\max} = \max\{\tau_i + t_i \mid v_i \in V\}.$$

Пусть для каждого задания известна неубывающая функция штрафа $\varphi_i(t)$. Обозначим $\overline{t_i}$ время окончания выполнения задания v_i в расписании S, тогда штраф при выполнении задания v_i равен $\varphi_i(\overline{t_i})$.

Можно выделить две большие группы задач: целевая функция $F_{\max}(S) = \max\{\varphi_i(\overline{t_i}) \mid v_i \in V\}$ называется максимальным штрафом, а $F_{\sum}(S) = \sum_{i=1}^{n} \varphi_i(\overline{t_i})$ — суммарным штрафом.

Требуется составить расписание, минимизирующее максимальный или суммарный штраф. Если для каждого задания определен директивный срок D_i — момент времени, к которому оно должно быть выполнено, то в качестве функции штрафа часто выбирается:

- функция временного смещения $\varphi_i(t) = t - D_i$;
- функция запаздывания $\varphi_i(t) = max\{0, t - D_i\}$.

Классификация задач составления распиний

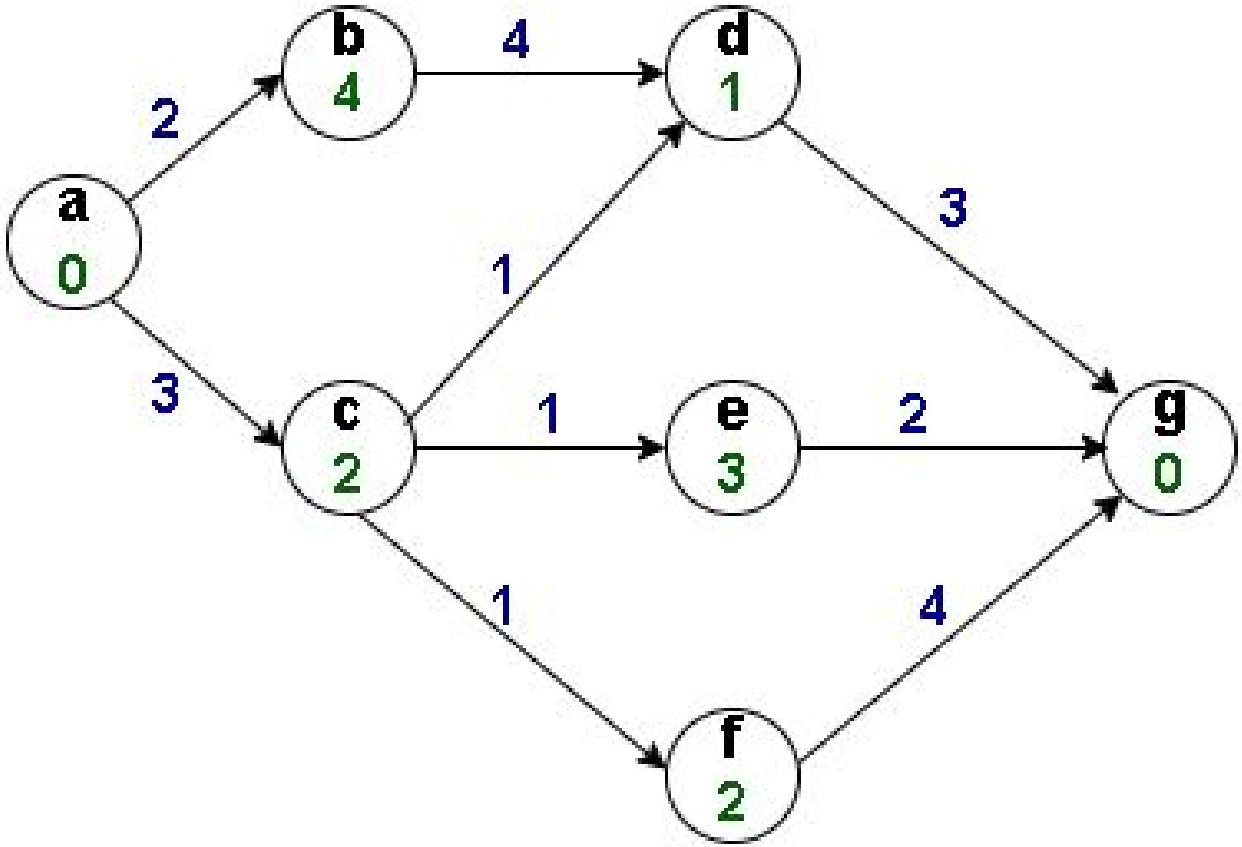
Можно выделить следующие группы задач:

1. 1. Задачи с параллельными процессорами
2. 2. Задачи конвейерного типа
3. 3. Однопроцессорные задачи
4. 4. Циклические задачи составления расписаний
5. 5. Задачи с учетом затрат на передачу данных
6. 6. Он лайн расписания

Примеры задач составления расписаний

1. Задача Джонсона (полиномиально разрешимая задача). Имеем конвейер из двух машин: m=2. Каждая работа состоит из двух операций с длительностями и , минимизируется общее время обслуживания работ.

Задача с учетом затрат на передачу данных



1	a	b					e					
2		c						f		g		
3						d						
t	0	1	2	3	4	5	6	7	8	9	10	11

Примеры задач составления расписаний

2. П р и м е р. Рассмотрим три задания 1, 2, 3, которые выполняются на четырех процессорах A, B, C, D. Последовательности процессоров для каждого задания (маршруты) приведены в таблице.

k	P	t(k,P)	P	t(k,P)	P	t(k,P)	P	t(k,P)
1	A	5	B	6	D	4	—	—
2	A	6	—	—	—	—	—	—
3	A	4	C	6	B	3	D	6

Задание 1 состоит в последовательном выполнении операций $(1, A)$, $(1, B)$ и $(1, D)$. Задание 2 состоит из одной операции $(2, A)$ и задание 3 из операций $(3, A)$, $(3, C)$, $(3, B)$ и $(3, D)$.

Графическое представление задачи

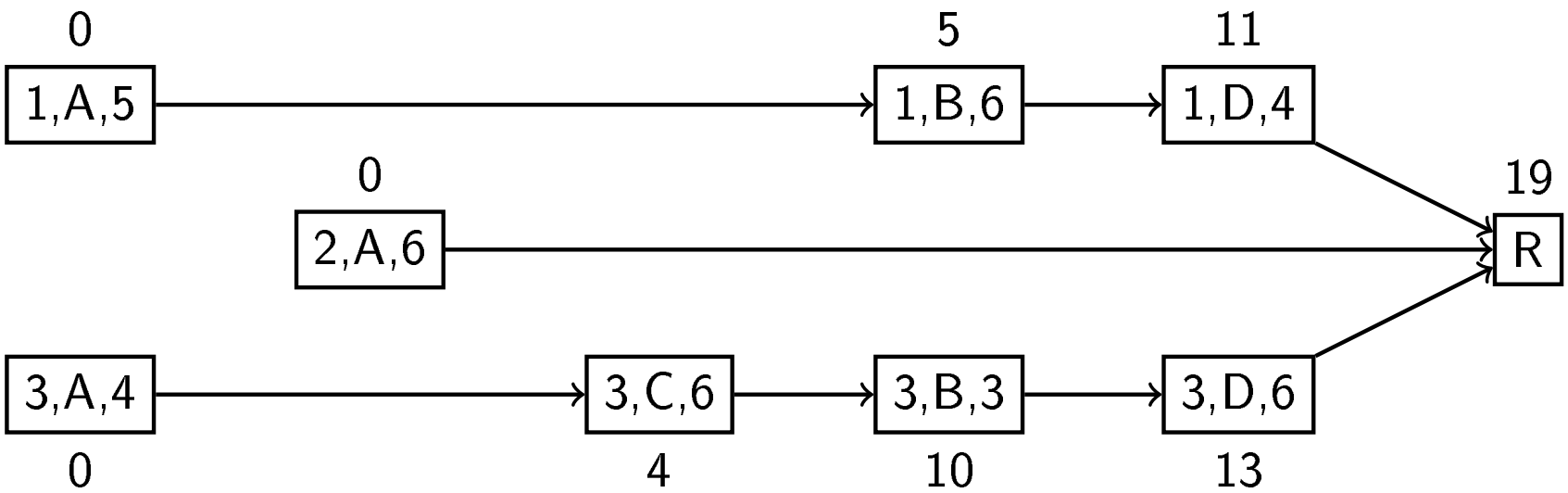


Figure:

Смешанный граф

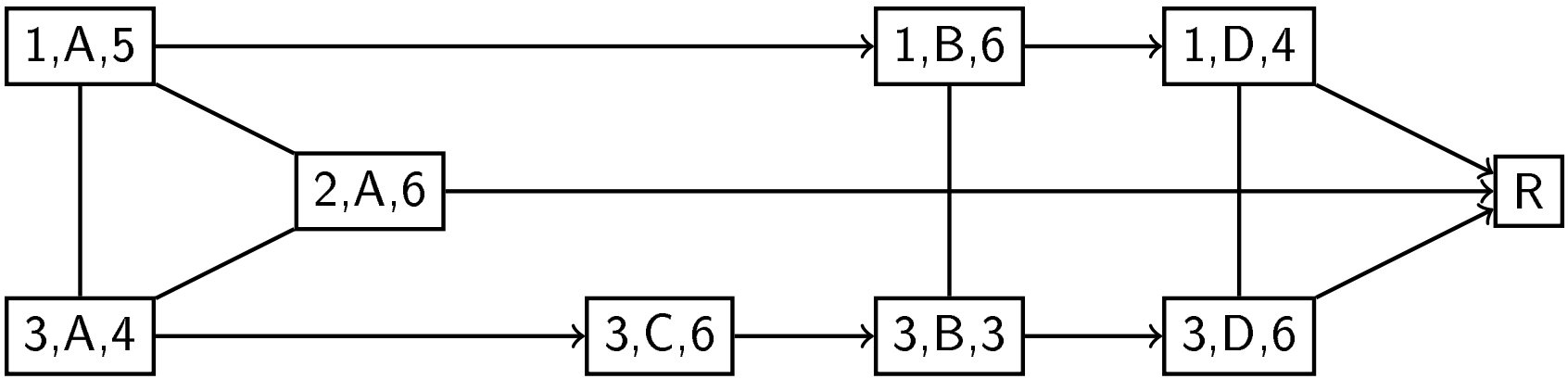


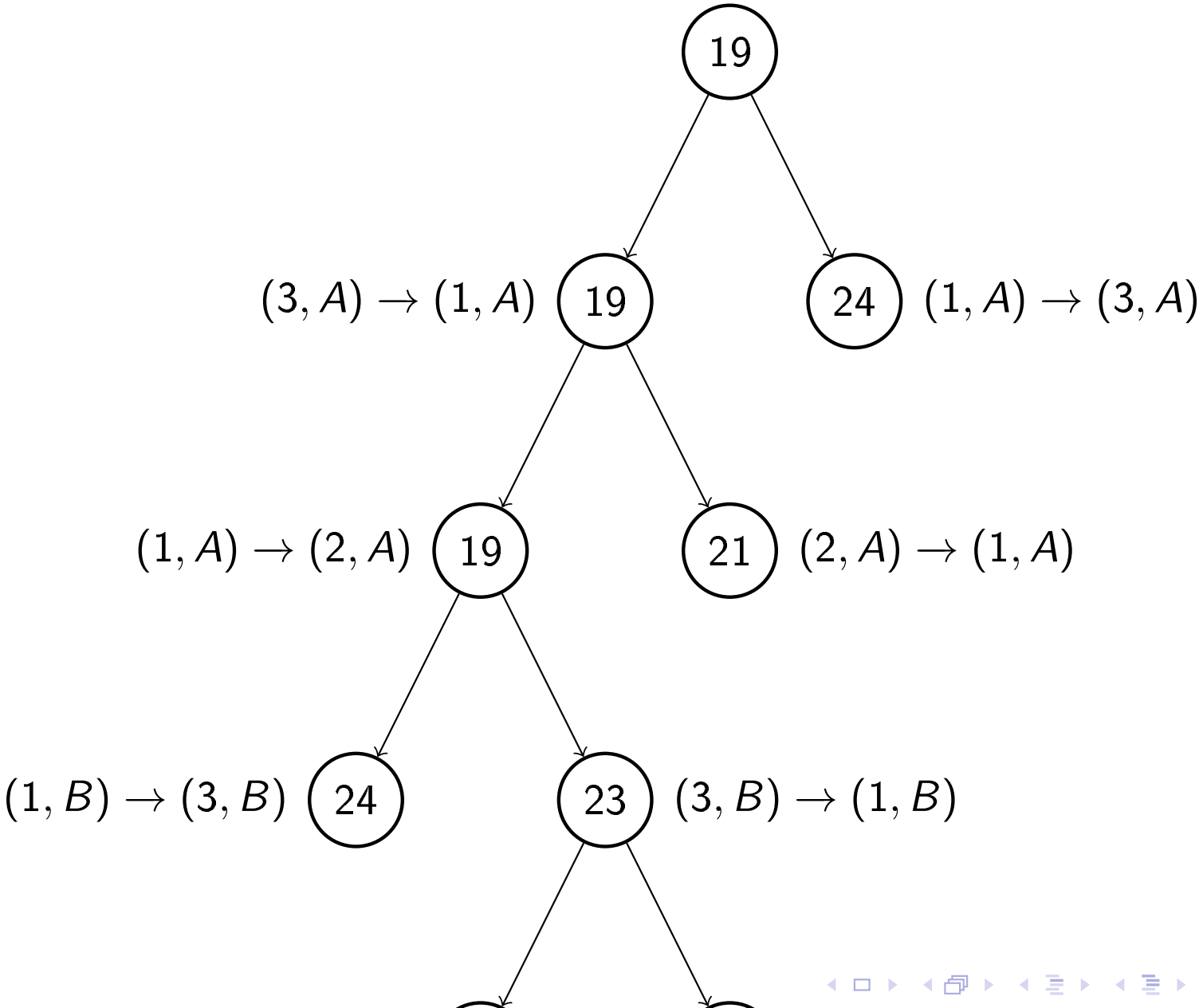
Figure: Смешанный граф

Метод ветвей и границ

Результаты работы метода ветвей и границ сведем в таблицу:

Итерация	Дуга	est	Выбор	Комментарий
1	(3,A) (1,A)	19	выбрали	*
1	(1,A) (3,A)	24	запомнили	*
2	(1,A) (2,A)	19	выбрали	*
2	(2,A) (1,A)	21	запомнили	*
3	(1,B) (3,B)	24	запомнили	*
3	(3,B) (1,B)	23	выбрали	*
4	(3,D) (1,D)	23	выбрали	записали рекорд 23
4	(1,D) (3,D)	25	запомнили	*
Вернемся к частичному решению с оценкой 21 на итерации 2				
5	(3,A) (2,A)	25	запомнили	Обе оценки частичных решений
5	(2,A) (3,A)	25	запомнили	превосходят рекорд 23

Метод ветвей и границ



Процесс продолжается до тех пор, пока не окажутся рассмотренными все ребра смешанного графа. В результате получаем бесконтурный ориентированный граф. Этот граф вместе с информацией о временных длительностях операций представляет собой один из допустимых сетевых графиков выполнения операций.

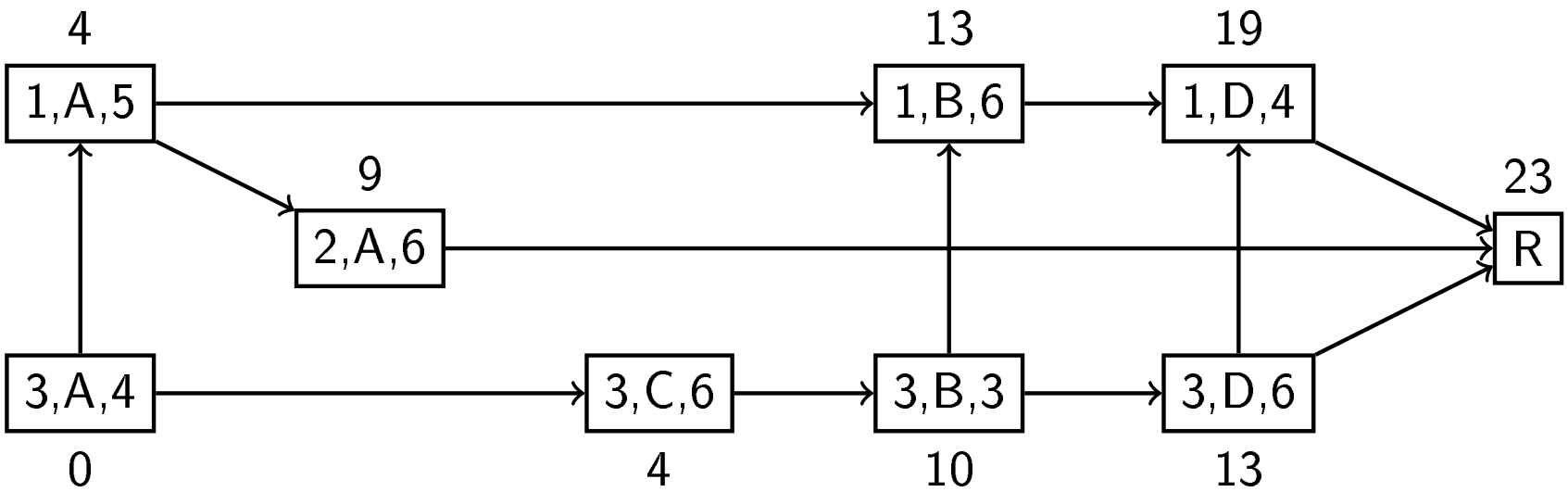


Figure: Сетевой график

Методы решения задач составления расписаний

«Временной» характер задач ТР выделяет их в особый класс задач математического программирования. В задачах ТР необходимо определить, когда, в какой последовательности выполнять работы. Это различие в существе задач определяет различие в методах и возможностях их решения. Для задач объемного характера развит достаточно мощный аппарат, главным образом математического программирования, позволяющий, в общем, с успехом добиваться их решения. Для задач ТР решающий аппарат развит в гораздо меньшей степени. Поиск оптимального или близкого к оптимальному расписания осуществляется с помощью одного из подходов:

1. - математического программирования;
2. - метод ветвей и границ;
3. - комбинаторного;
4. - эвристического;
5. - статистического (вероятностного).

Методы решения задач составления расписаний

Формулировка общей задачи составления расписания в терминах линейного целочисленного программирования

Комбинаторный подход

Комбинаторный подход
Вспомним сначала определения таких понятиях, как задачи класса P, эффективные алгоритмы и NP-полные (трудные) задачи. Под «эффективным алгоритмом» понимается алгоритм, для которого число требуемых шагов растет как полином от размера входной задачи. Задачи, имеющие эффективные (полиномиальные) алгоритмы решения, принадлежат классу P-задач. Класс NP-задач обладает следующими свойствами: - для никакую NP-полную задачу нельзя решить никакими известными полиномиальными алгоритмами; - если существует полиномиальный алгоритм для какой-нибудь NP-полной задачи, то существуют полиномиальные алгоритмы для всех NP-полных задач. Практическое значение понятия NP-полноты состоит в следующем: такие задачи по существу труднорешаемы с вычислительной точки зрения, они не поддаются эффективному алгоритмическому решению.

Метод ветвей и границ

Метод ветвей и границ (и различные его модификациии) это метод, который позволяет получить оптимальное решение задачи дискретной оптимизации, но для этого может потребоваться экспоненциальное от размера задачи количество времени. Для некоторого ограниченного круга задач оптимальное решение может быть получено методом динамического программирования (который является псевдополиномиальным алгоритмом).

Эвристические алгоритмы

Эвристические алгоритмы быстро строят решение, не очень плохое. Эвристические алгоритмы используют различные разумные соображения без строгих обоснований, которые не гарантируют получение оптимального решения, но часто генерируют вполне приемлемые, с практичексой точки зрения, расписания.

Списочные алгоритмы

Особую роль в ТР играют списочные алгоритмы. Списочный алгоритм состоит из двух этапов:
1. Заданиям присваиваются приоритеты.
2. Каждый раз, когда процессор освобождается, на него назначается готовое задание с максимальным приоритетом.

1. 1) Правило SPT (shirtest processing time). Предпочтение отдается той работе (операции) из множеств, у которой время выполнения минимально.
2. 2) Правило LRT (longest remaining time). Требует выбора напряженной работы, т.е. той, у которой сумма времен выполнения оставшихся операций наибольшее.
3. 3) Правило LPT (longest processing time). Предпочтение отдается той работе (операции) , у которой время выполнения максимально.
4. 4) FIFO — генератор, использующий правило “первый пришел — первый обслуживается”, (First In — First On).
5. 5) LIFO — генератор, использующий правило “последний пришел — первый обслуживается”, (LAST In — First On).

Алгоритмы с гарантированной оценкой точности

При оценке эвристического алгоритма рассматривают два критерия: качество расписания и вычислительная сложность (трудоемкость) алгоритма. Говорят, что алгоритм H, решающий задачу D, имеет гарантированную оценку точности равную σ

$$\sigma = \inf\{r \geq 1 : f_A(I)/f_{opt}(I) \leq r\}\},$$

где inf берется по всему множеству примеров задачи D;
I — конкретный пример;
$f_A(I)$ - значение целевой функции, полученное алгоритмом H для примера I;
$f_{opt}(I)$ —оптимальное значение целевой функции для примера I.

Метаэвристики

Вероятностные методы связаны с многократным моделированием расписаний.
Широко применяется так называемый метод локального поиска. Выбирается начальное решение и набор допустимых преобразований этого решения. Текущие решение преобразуется операциями из данного набора. Если удается получить лучшее решение, то алгоритм переходит в это решение и процесс продолжается.
После k-кратного проигрывания выбирается наилучшее расписание, которое принимается за решение задачи. Существуют разные алгоритмы, основанные на идее локального поиска.

1. а) поиск в локальной окрестности;
2. б) направленный случайный поиск с самообучением;
3. в) имитация отжига;
4. г) поиск с запретами.

Другие подходы реализуют следующие алгоритмы:

1. д) генетические алгоритмы;
2. е) нейронные сети;

Литература

- Graham, R.L., Lawner, E.L., Rinnoy Kan,A.H.G.: Optimization and approximation in deterministic sequencing and scheduling. A survey. Ann. of Disc. Math. **5** (10), 287–326 (1979)
- Brucker Scheduling Algoritms. (fifth edition) Springer, New York 2007
- Pinedo M. Scheduling: Theory, Algorithms, and Systems. 2014

Однопроцессорная задача составления расписания с временами поступления и доставки

Григорьева Наталья Сергеевна

СПбГУ

2023

Введение

- Problem Statement $1|r_j, q_j|C_{\max}$.
- Approximation algorithm IJR.
- The worst-case performance ratio of algorithm IJR.
- Метод ветвей и границ.

Постановка задачи

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 $r(i) \leq \tau(i)$. Характеристика качества расписания — время доставки последней работы $C_{\max} = \max\{\tau(i) + t(i) + q(i) | i \in U\}$. Требуется построить расписание минимизирующее $C_{\max}$.

Обзор

Задача NP-трудна в сильном смысле, но существуют точные полиномиальные алгоритмы для ряда специальных случаев.

- Первым алгоритмом построения приближенного расписания была эвристика Шраге [2]— расширенное правило Джексона, которое формулируется следующим образом: каждый раз, когда процессор освобождается, на него назначается готовая работа с максимальным временем доставки
  Вычислительная сложность алгоритма $O(n \log n)$. В работе Х. Кизе и др. показано, что алгоритм имеет гарантированную оценку точности, равную 2.
- К. Поттс [3] 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 расписание. Вычислительная сложность алгоритма Поттса $O(n^2 \log n)$. Гарантированная оценка точности

Обзор

- Л. Холл и Д.Шмойс [4] 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 задачи. 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 [5] 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 множество устанавливаются в порядке поступления, из второго – по невозрастанию времени доставки.
- Автором [3] был предложен алгоритм, который строит две перестановки: одну алгоритмом Джексона, а другую алгоритмом с простоями IJR и выбирает из них лучшее. гарантированная оценка точности 3/2.

Сводная таблица

Table: Approximation algorithms

Year	P_n	Comp. c	WCR	Ref.
1971	1	$O(n \log n)$	2	Schrage
1980	n	$O(n^2 \log n)$	3/2	K.Potts
1992	$4n$	$O(n^2 \log n)$	4/3	L.Hall and D. Schmois
1994	2	$O(n \log n)$	3/2	E. Novitsky and K. Smutnitsky
2021	2	$O(n \log n)$	3/2	N. Grigoreva

Цель доклада показать. что алгоритм IJR имеет гарантированную оценку точности 11/7.

Основная идея алгоритма

Основная идея алгоритма IJR : 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. Эти условия позволяют сделать выбор между двумя заданиями.

Алгоритм IJR

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

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

Отсортируем задания по не убыванию времен поступления $r(j_1) \leq r(j_2) \leq \ldots \leq r(j_n)$. 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 поставлены в расписание, S_{k-1} – подмножество заданий, включенных в расписание. Время освобождения процессора известно $time := \max\{\tau(i) + t(i) | i \in S_{k-1}\}$. Опишем $k-$ ый шаг.

Алгоритм IJR

1. Добавляем в очередь Q_1 готовые задания такие, что $r(i) \leq time$.
   Если готового задания в момент времени $time$ нет, то
   $time := \min\{r(i) | i \notin S_{k-1}\}$ и переходим к пункту 1.
2. Выбираем готовое задание $u \in Q_1$ с максимальным временем доставки
   $q(u) = \max\{q(i) | i \in Q_1\}$.
3. Положим $r_{up} := time + t(u)$.
4. 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 пункту 8.
5. Если выполнено $q(j_i) > LB/2$, то для задания j_i определяем возможный простой процессора $idle(j_i) = r(j_i) - time$.
   Проверяем условие $q(j_i) - q(u) \geq idle(j_i)$.

Алгоритм IJR

1. Если условия 5 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 пункту 1. Иначе добавляем j_i задание в очередь.
2. Если $r(j_{i+1}) < r_{up}$, то положим $i := i + 1$, перейдем к пункту 5.
3. Если не нашли задание u^*, то устанавливаем на процессор задание u, положим $\tau(u) := time$; $time := \tau(u) + t(u)$.
   Если $k < n$, то $k := k + 1$, переходим к 1, иначе конец алгоритма, расписание S_n построено.

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

Комбинированный алгоритм построения расписания ICA.

Комбинированный алгоритм построения расписания ICA.
1. Строим расписание S_{JR} алгоритмом Шраге (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: $C_{\max}(S_A) = \min\{C_{\max}(S), C_{\max}(S_{JR})\}$.

Гарантированная оценка точности

Theorem

Алгоритм строит расписание S_A для которого
$C_{\max}(S_A)/C_{\max}(S_{opt}) \leq 3/2$.
Вычислительная сложность алгоритма $O(n \log n)$.*[3]*

Пример

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

Таблица 1.

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 $S_{opt} = (x, u, a, c)$, значение целевой функции для которого $C_{\max}(S_{opt}) = 2 * \varepsilon + M/2 + \varepsilon + M/2 - 2 * \varepsilon = M + \varepsilon$.

При ε стремящимся к нулю отношение $C_{\max}(S_A)/C_{\max}(S_{opt})$ стремится к 3/2.

The worst-case performance of algorithm IJR

Пусть C_A длина расписания. построенного алгоритмом IJR и C_{opt} длина оптимального расписания.

Theorem

Алгоритм IJR генерирует расписание S_A, для которого $C_A/C_{opt} < 11/7$.
[5]

IJR алгоритм строит одну перестановку и имеет вычислительную сложность $O(n \log n)$.

Lemma

Пусть алгоритм IJR генерирует расписание S, значение целевой функции равно C_A.
Пусть существует интерференсная работа u в критической последовательности $J(S) = (a, S_1, u, S_2, c)$. Если в оптимальном расписании работа u выполняется после последовательности S_2 и работы c или между работами a и c тогда $C_A/C_{opt} \leq 3/2$.

The worst-case performance of algorithm IJR

Lemma

Пусть имеются задержанные работы в критической последовательности $J(S) = (a, S^*, c)$. *но нет интерференсной работы, тогда тогда* $C_A/C_{opt} \leq 3/2$.

Задача линейного программирования

Нам осталось расссмотреть случай , в котором в оптимальном расписании интерференсная работа u должна выполняться перед работой a. Этот сдучай можно формализовать в виде задачи целочисленного линейного программирования.

$$F(\mathbf{y}) = y_1 + y_2 + y_3 + y_4 + y_6 + y_9 \longrightarrow \max$$

$$\begin{aligned}
y_2 + y_3 + y_4 + y_9 - LBw &\le 0 \\
y_1 + y_5 + y_9 - LBw &\le 0 \\
y_1 + y_2 - y_7 + y_9 + y_{11} &\le 0 \\
y_6 + y_7 + y_{10} - LBw &\le 0 \\
y_3 + y_6 + y_7 &\le 1 \\
y_4 + y_5 + y_9 &\le 1 \\
y_3 + y_4 + y_6 + y_9 &\le 1 \\
-y_5 + (LB/2 + 1)w &\le 0 \\
-y_9 + w &\le 0 \\
-y_{10} + w &\le 0
\end{aligned}$$

Задача линейного программирования

Задача линейного программирования решается в общем виде. Наихудший случай работы алгоритма имеет место в случаи, когда алгоритм создает большой простой процессора, в то время как в оптимальном расписании задания выполняются без простоя. Мы доказали [3], что

$$\max C_A/C_{opt} = (11LB - 14)/(7LB + 2), LB - even.$$

$$\max C_A/C_{opt} = (11LB - 13)/(7LB + 1), LB - odd.$$

LB — нижняя оценка целевой функции в исходной задаче.

Example

Рассмотрим систему из 21 работы : a, u, S_1, c и 17 одинаковых работ, которые включены в S_2. Исходные данные приведены в Table.

Table: Release, processing and delivery times of jobs

Job	a	u	S_1	j_1	$\ldots$	j_{17}	c
r_i	48	0	48	65	65	65	65
t_i	1	65	16	1	1	1	1
q_i	51	0	1	33	33	33	33

Нижняя граница целевой функции равна $LB = 100$.

Example

Алгоритм IJR генерирует расписание $S = (a, S_1, u, j_1, \ldots, j_{17}, c)$. Процессор простаивает i 48 единиц времени перед началом работы a. Работа c является критической работой, выполнение работы c заканчивается в момент времени 148 и доставка работы заканчивается в момент времени 181.

IJR schedule

48	1	16	65	1	16	1	1	33
idle	a	S_1	u	j_1	$\ldots$	j_{17}	c	$q(c)$

Целевая функция равна
$C_{\max}(S) = LB/2 - 2 + 1 + (LB - 4)/6 + (LB + 8)/6 + (2LB - 5)/3 + (LB - 1)/3 = (11LB - 14)/6 = 181.$

Оптимальное расписание

Оптимальное расписание

65	1	17	1	16
u	a	S_2	c	S_1

Оптимальное расписание is $S_{opt} = (u, a, S_2, c, S_1)$, значение целевой функции равно $C_{opt} = 6/(7LB + 2) = 117$.
$C_A/C_{opt} = (11LB - 14)(7LB + 2) = 1.547$.

Branch and Bound algorithm [4]

Root node.

Нижняя граница Найти расписание для задачи, в котором разрешены прерывания и найти LB. Если в расписании не оказалось прерванных работ, то построенное расписание оптимально.

Schrage's heuristic Найти расписание S_r алгоритмом Джексона и обозначить $C_{\max}(S_r)$ его длину. Если $UB > C_{\max}(S_r)$ тогда $UB = C_{\max}(S_r)$. Если $LB = UB$, то расписание оптимально и алгоритм заканчивается.

IJR heuristic

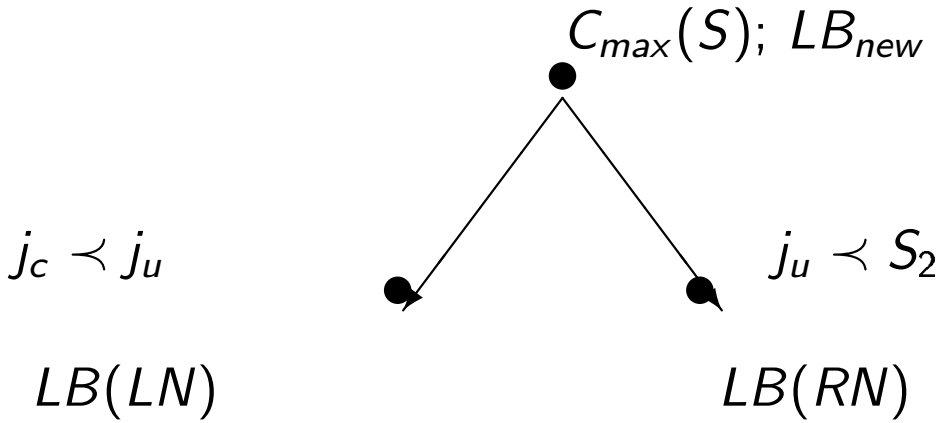
Найти расписание S алгоритмом IJR и обозначим $C_{max}(S)$ его длину. Если $UB > C_{\max}(S_r)$ то $UB = C_{max}(S)$. Если $LB = UB$, то расписание S оптимально и алгоритм заканчивается.

Branch and Bound algorithm

Branch. Найти критическую последовательность $J(S)$ и проранализировать ее.
Если нет интерференсных и задержанных работ, то расписание S оптимально S и алгоритм заканчивается.

Algorithm

Есть интерференсная работа j_u в $J(S)$.



$$LB_{new} = max(LB, C_{max}(S) - t(j_u) + idle).$$

$$LB(LN) = \max(LB_{new}, C_{\max}(S) + \delta(S) - q(j_c) + q(j_u)).$$
$$LB(RN) = \max(LB_{new}, C_{\max}(S) - \tau(j_u) + r(j_u).$$

Algorithm

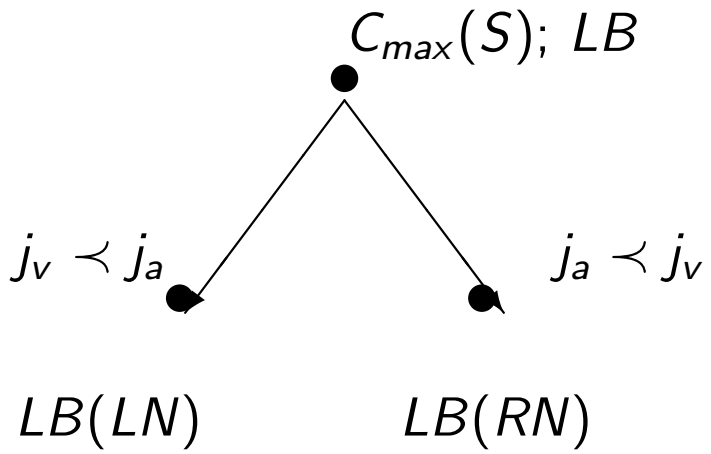
$LB = max(LB, C_{max}(S) - t(j_u) + idle)$. Если $LB \geq UB$, то текущая вершина закрывается, в противном случае создаем левую и правую вершину.

Algorithm

Right node.В правой вершине дерева перебора работа job $j(u)$ должна выполняться до последовательности S_2 and условие $j_u \prec S_2$ добавляется к ограничениям.
$LB = \max(LB, C_{\max}(S) - r(j_a) - T(S_1) + r(j_u))$.
$q(j_u) = q(j_c) + T(S_2)$ и пересчитываются $q(j)$ для всех предшественников (j_u).
Строится новое расписание S, $UB = \min(UB, C_{max}(S))$. Если $LB \geq UB$, то текущая вершина не может улучшить целевую функцию и она закрывается.
Left node. В левой вершине условие $j_c \prec j_u$ добавляется.
$LB = \max(LB, C_{\max}(S) - \delta(S) + q(j_c) - q(j_u))$.
If $LB \geq UB$, then the left node is eliminated else set $r(j_u) = r(j_c) + t(j_c)$ и пересчитывается $r(j)$ для всех потомков of job j_u.
Строится новое расписание S, $UB = \min(UB, C_{max}(S))$. Если $LB \geq UB$, то текущая вершина не может улучшить целевую функцию и она закрывается.

Algorithm

Существует задержанная работа j_v в $J(S)$.



$$LB(LN) = max(LB, C_{max}(S) + r(j_v) - r(j_a)).$$

$$LB(RN) = max(LB, C_{\max}(S) - T(J(S)) + t(j_v) + T(J_1)).$$

$$LB = max(LB, C(S) - r(j_u) + r(j_a)).$$

Branch and Bound algorithm

$LB = \max(LB, C_{\max}(S) + r(j_a) - r(j_v))$.
Left node. В левой вершине добавляется условие $j_v \prec j_a$. Пересчитывается LB
$LB = \max(LB, C_{\max}(S) + r(j_a) - r(j_v))$.
Если $LB \geq UB$, то левая вершина закрывается, иначе $q(j_v) = q(j_a) + t(j_a)$ и пересчитываются $q(j)$ для всех предшественников (j_v).
Right node. В правой вершине условие $j_a \prec j_v$ добавляется.
$LB = max(LB, C_{\max}(S) - T(J(S)) + t(j_v) + T(J_1))$.
Если $LB \geq UB$, то правая вершина закрываеся, иначе
$r(j_v) = r(j_a) + t(j_a)$ пересчитываем $r(j)$ для всех потомков j_v .
Строим новое расписание S, полагаем $UB = \min(UB, C_{max}(S))$.
Алгоритм работает пока есть незакрытые вершины.

Computational Results

Число работ варьировалось от $n = 50$ до $n = 5000$.
Время выполнения генерировалось равномерно в диапазоне от 1 до 50.
Времена поступления и доставки генерировались равномерно из диапазона от 1 до Kn, для $K = 20$. Pan показал (2006) что наиболее трудные примеры встречаются при $14 \leq K \leq 25$.
Для каждого n и $K = 20$ генерировалось 100 примеров.

Computational Results

Первая колонка - число работ n.
Колонка V, содержат средние значения числа вершин в дереве перебора, VM максимальное число верщшин в дереве перебора и T среднее время вычислений (в сек), для *BBI* . Колонки VC, VMC, TC содержат те же самые значения для алгоритма Карлье [6], а VH, VMH, TH содержат те же самые значения для алгоритма Чандры [1].

Computational Results

Table: Performance of algorithms according to the variation of n

n	V	VC	VH	VM	VMC	VMH	T	TC	TH
50	1.05	10.7	1.5	2	17	24	0.00	0.00	0.00
100	1.00	15.8	1.4	1	28	3	0.00	0.00	0.00
300	1.06	28.4	1.3	2	188	59	0.00	0.07	0.01
500	1.15	46.2	1.5	2	115	19	0.01	0.35	0.04
1000	1.05	52.5	1.5	2	220	4	0.04	1.79	0.41
2000	1.10	34.0	1.5	2	77	3	0.20	5.14	0.84
5000	1.05	59.4	1.5	2	146	3	1.38	72.84	3.23

Литература

- Chandra, C.,Liu, Z.,He, J., Ruohonen, T.: A binary branch and bound algorithm to minimize maximum scheduling cost. Omega **42**, 9–15 (2014)
- Schrage L. : Optimal Solutions to Resource Constrained Network Scheduling Problems (unpublished manuscript) 1971.
- Potts, C.N.: Analysis of a heuristic for one machine sequencing with release dates and delivery times. Operational Research. **28** (6), 445–462 (1980)
- Hall, L.A. and Shmoys, D.B.: Jackson's rule for single-machine scheduling: making a good heuristic better// Mathematics of Operations Research, 1992, **17** (1), 22—35 (1992)
- Nowicki E., Smutnicki C. :An approximation algorithm for a single-machine scheduling problem with release times and delivery times. Discrete Applied Mathematics , **48**, 69–79 (1994)
- Carlier, J.: The one machine sequencing problem. European Journal of Operational Research. **11**, 42–47 (1982)

Литература

- Pan, Y., Shi, L.: Branch and bound algorithm for solving hard instances of the one-machine sequencing problem. European Journal of Operational Research, **168**, 1030—1039 (2006)
- Liu, Z., Single machine scheduling to minimize maximum lateness subject to release dates and precedence constraints. Computers & Operations Research **37**, 1537-1543 (2010),
- Grigoreva, N.S.: Worst-Case Analysis of an Approximation Algorithm for Single Machine Scheduling Problem In : Proceedings of the 16th Conference on Computer Science and Intelligence Systems, vol. 25, pp. 221-225. (Annals of Computer Science and Information System; v. 25).(2021)
- Grigoreva N.S.: Single Machine Scheduling with Precedence Constrains, Release and Delivery times. Advances in Intelligent Systems and Computing; V. 1052, pp. 188 —198, (2019)
- N.Grigoreva : An 11/7- approximation algorithm for single-machine scheduling problem with release and delivery time. Advances in Optimization and Applications. Communications in Computer and Information Science (CCIS) V 1739, pp. 76—89 (2022)