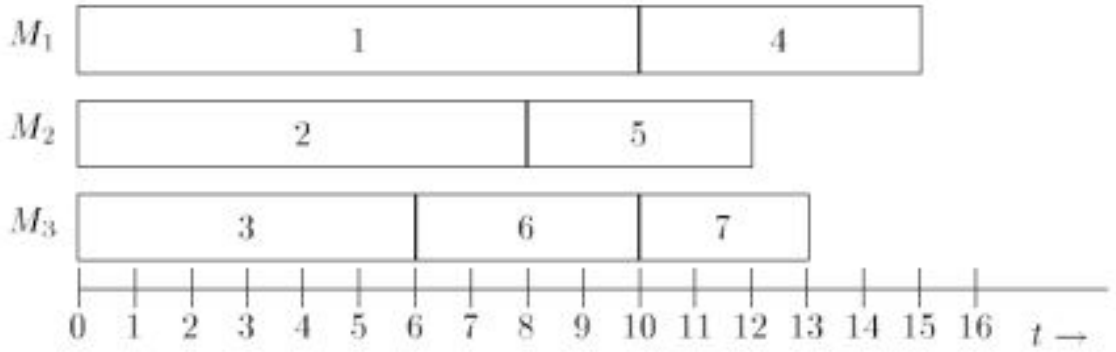
Простейшие задачи теории расписаний

Задачи, возникающие у мастера в автомастерской,
у машинистки с подготовкой рукописей,
в центральном процессоре с исполнением заданий

Задачи построения оптимальных расписаний

- Имеется несколько станков/рабочих/компьютеров M_j
- Имеется набор деталей/задач/программ, требующих обработки
- Требуется: распределить задачи по рабочим так, чтобы время окончания обработки последней задачи было минимальным



Основные обозначения

Рассматриваются два множества:

$M = \{M_1, M_2, \ldots, M_m\}$ — машины (станки, процессоры, бригады, …)

$J = \{J_1, J_2, \ldots, J_n\}$ — работы (задания, пакеты задач, …)

- ***Расписание*** — указание, на каких машинах и в какое время должны выполняться работы.

В каждый момент времени каждая машина выполняет не более одной работы, и каждая работа выполняется на одной машине или не выполняется вовсе.

Работы состоят из операций: $J_i = \{O_{i_1}, O_{i_2}, \ldots, O_{i_{n_i}}\}$

Операция O_{i_j} требует p_{ij} времени и может выполняться на одной из машин множества $\mu_{ij} \subseteq \{M_1, \ldots, M_m\}$.

Если $|\mu_{ij}| = 1, \forall\, i\, j,$ то получаем модель с предписаниями.

Если $|\mu_{ij}| = m, \forall\, i\, j,$ то получаем модель с параллельными машинами.

Для работы J_i известны:

$r_i \geq 0$ — время появления первой операции O_{i_1}

$d_i \geq 0$ — директивное время окончания последней операции $O_{i_{n_i}}$

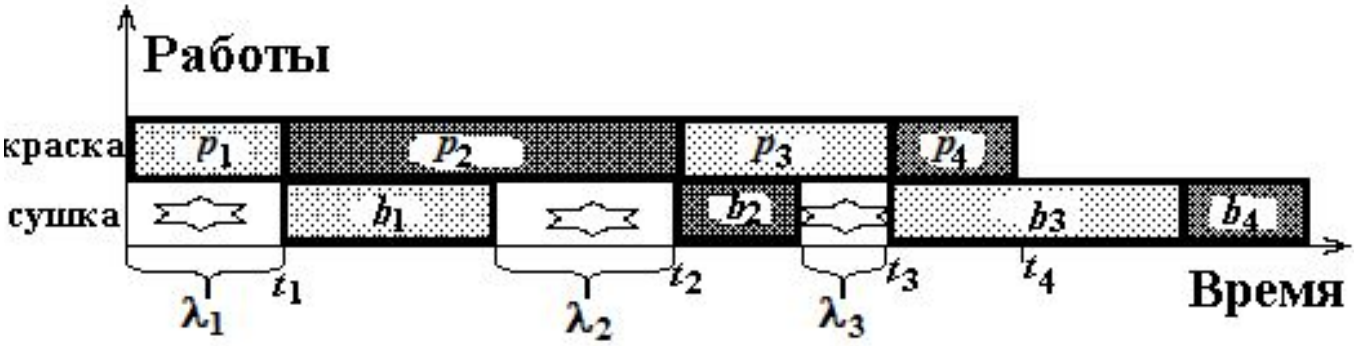
$w_i \geq 0$ — важность (вес, ценность) работы J_i

Задачи одного исполнителя на минимакс:

- Исполнитель должен последовательно выполнить n работ
- Каждая j-я работа имеет известную длительность p_j
- Работы можно выполнять в любом порядке и начинать в произвольный момент времени
- Каждая работа выполняется с самого начала и должна быть выполнена полностью прежде чем начнется выполнение следующей
- Разрывы в исполнении работ невыгодны
- Чем скорее будет выполнена любая работа, тем лучше (задача на минимакс)
- Время окончания работ t_j зависят от перестановки π (порядок выполнения работ)
- $T\{ t_1(\pi), t_2(\pi),\ldots, t_n(\pi)\}$ – называется расписанием

Задача красильщика

Красильщик должен окрасить *n* предметов. Для окраски *j*-го предмета требуется время p_j. По окончании окраски *j*-й предмет должен еще сохнуть в спец. камере время b_j после чего он считается готовым. В каком порядке нужно красить предметы, чтобы последний в просушке был готов как можно раньше?



Каждой *j*-ой работе можно сопоставить штрафную функцию

$$\psi(t) = t_j + b_j$$

Задача: Построить такое расписание $T(\pi)$, чтобы минимизировать $f = \max(\psi_j)$

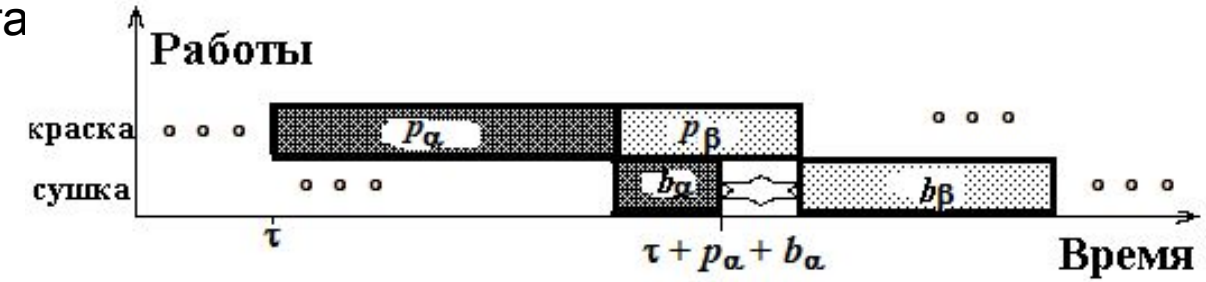
Теорема 1: Оптимальное расписание в задаче красильщика получается упорядочением работ по убыванию b_j

Доказательство: Пусть **π' –** перестановка, для которой выполняется свойство: $b_{\pi'(1)} \geq b_{\pi'(2)} \geq \ldots \geq b_{\pi'(n)}$. Предположим противное: **π'** не является оптимальной
Тогда существует некоторая оптимальная перестановка π_0 такая, что

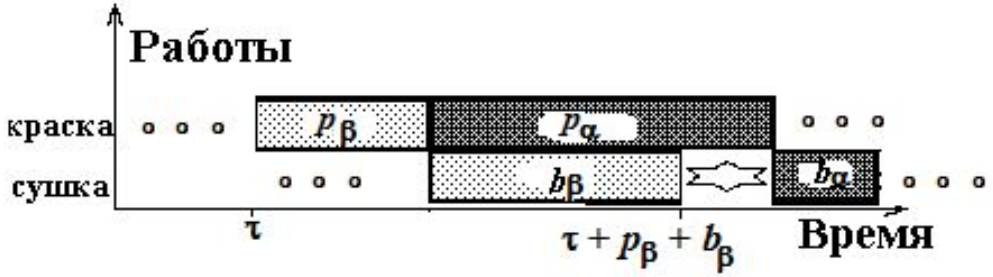
$$f(T(\pi_0)) < f(T(\pi_0))$$

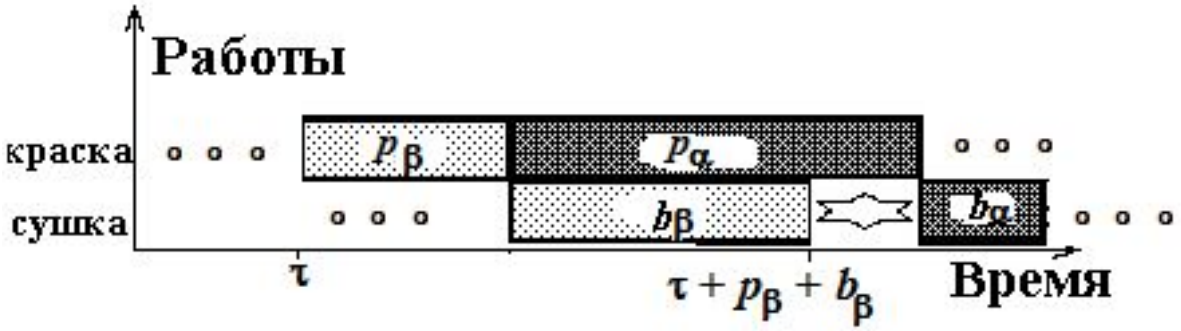
Так как $\pi_0 \neq \pi'$, то в перестановке π_0 найдется пара работ **α** и **β** таких, что

$\alpha \propto \beta$, но $b_\alpha < b_\beta$. Пусть суммарная длительность работ, стоящих в переста



Тогда, учитывая, что $b_\alpha < b_\beta$: $\max(\tau + p_\alpha + b_\alpha , \tau + p_\alpha + p_\beta + b_\beta) = \tau + p_\alpha + p_\beta + b_\beta$
Составим перестановку π_1 , в которой работы **α** и **β** переставлены местами:





тогда вклад в ***f*** от работы **β** равен $\tau + p_\beta + b_\beta$ а от работы **α** равен: $\tau + p_\beta + p_\alpha + b_\alpha$

Можно заметить: $\tau + p_\beta + b_\beta < \tau + p_\alpha + p_\beta + b_\beta$ (что является max для π_0) т.к. $p_\alpha > 0$;

и одновременно $\tau + p_\beta + p_\alpha + b_\alpha < \tau + p_\alpha + p_\beta + b_\beta$, т.к. $b_\alpha < b_\beta$

Поэтому $f(\pi_1) \le f(\pi_0)$. Если в перестановке π_1 присутствует еще одна такая пара, то с ней можно поступить аналогично и т.д. В результате получим перестановку в которой выполняется свойство: $b\pi'(1) \ge b\pi'(2) \ge \ldots \ge b\pi'(n)$. Ч.т.д.

«Небольшое» усложнение задачи: обычно не все работы известны в начальное время, они могут появляться в моменты времени r_j. Таким образом, работе j сопоставлен не только «хвост» b_j, но и возможно ранние начала r_j.

Такая задача *может быть решена только «перебором»*!!!

В Календарном планировании каждой работе сопоставлен директивный срок выполнения d_j. *Отклонением* от директивного срока называют величину:

$$l_j(T(\pi)) = t_j(T) - d_j$$

В различных задачах могут рассматриваться штрафные функции для отклонений как положительных, так и отрицательных.

Задача Джексона (1954 – 1955 г.)

Рассматриваются n работ, для каждой из которых определены длительности p_j и директивные сроки выполнения d_j. Построить такое расписание $T(\pi)$, чтобы минимизировать сумму отклонений $f = \sum (t_j(T) - d_j)$

Теорема 2: Оптимальное расписание T(π') в задаче Джексона достигается упорядочением работ по возрастанию их директивных сроков.

$$d_{\pi'(1)} \le d_{\pi'(2)} \le \ldots \le d_{\pi'(n)}$$

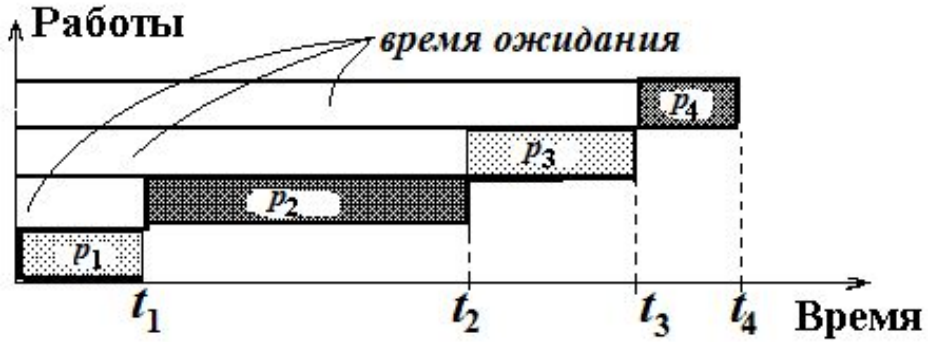
Док-во – аналогично док-ву Теоремы 1.

Алгоритм Лившица: Оптимальный порядок работ строится, начиная с конца. При любом порядке последняя работа окончится в момент $\boldsymbol{\omega} = \sum p_j$

Оценим значения штрафных функций задачи и найдем $\min \psi_j$. Пусть он достигается на k-й работе: $\min \psi_j = \psi_k$
Положим π'(n) = k и перейдем к задаче меньшего размера, полагая
$n = n - 1$; $\boldsymbol{\omega} = \boldsymbol{\omega} - p_k$; $J = J \setminus k$
Повторяя эту процедуру $\boldsymbol{n}$ раз, найдем оптимальную перестановку.

Задачи одного исполнителя на минисумму

Задача 1: Все n работ находятся в системе с момента $t = 0$. Каждая j-я работа Покидает систему в момент $t_j(T)$. Минимизировать среднее время пребывания работ в системе: $f(T) = \cdot \frac{1}{n} \sum t_j(T)$.



Теорема (Смит): Задача минимизации среднего времени решается упорядочением работ по возрастанию длительностей.

Задача 2: В условиях предыдущей задачи учитывается дополнительно различная стоимость c_j пребывания работ в системе. Минимизации подлежит следующий функционал: $f(\boldsymbol{\pi}) = \sum (c_j \cdot t_j)$.

Теорема (Мак-Нотон): Задача минимизации суммарного взвешенного времени пребывания работ в системе решается упорядочением работ по возрастанию параметра p_j / c_j.

Док-во этих теорем– аналогично док-ву Теоремы 1.

Решение аналогичных задач:

- Отклонение окончания j-й работы от директивного срока d_j : $l_j = t_j(T) - d_j$
- Минимизируемый функционал:

$$f_l = \sum(c_j \cdot l_j) = \sum(c_j \cdot (t_j - d_j)$$

- Задача решается алгоритмом Мак-Нотона, так как функционалы отличаются на константу

- Запоздание окончания j-й работы от директивного срока $dj : z_j = (t_j(T) - d_j)^+$
- Минимизируемый функционал:

$$f_l = \sum(c_j \cdot z_j) = \sum(c_j \cdot (t_j - d_j)^+$$

- Задача решается только перебором (NP-полная задача)

Случаи разрывной функции штрафа

Задача 1 (Мур): Минимизировать число запоздавших работ.

В этом случае штрафная функция выглядит так

$$\psi_j = \begin{cases} 0, & \text{если } t_j \le d_j \\ 1, & \text{если } t_j > d_j \end{cases}$$

Алгоритм Ходжсона:

1) Упорядочить все работы в порядке возрастания d_j (если $d_i = d_j$, то первой ставится работа с меньшим номером). В результате получим перестановку $\boldsymbol{\pi} = (\boldsymbol{\pi}(1), \boldsymbol{\pi}(2),\ldots, \boldsymbol{\pi}(k), \ldots, \boldsymbol{\pi}(n))$;
2) $k := 0$;
3) $k := k + 1$. Если работа k – отмечена или $k \ge n$, то полученная перестановка $\boldsymbol{\pi}'$ – **оптимальная. Конец.**
4) Если $t_{\boldsymbol{\pi}(k)} \le d_{\boldsymbol{\pi}(k)}$, то перейти к 3.
5) В случае, если $t_{\boldsymbol{\pi}(k)} > d_{\boldsymbol{\pi}(k)}$, найти из работ $\boldsymbol{\pi}(1), \boldsymbol{\pi}(2),\ldots, \boldsymbol{\pi}(k)$ работу самой большой длительности. Пусть это будет работа $j = \boldsymbol{\pi}(r)$. Поставить работу j в конец, а следующие за ней до $\boldsymbol{\pi}(k)$-й работы включительно сдвинуть к началу, то есть порядок $\boldsymbol{\pi}=(\boldsymbol{\pi}(1), \boldsymbol{\pi}(2),\ldots, \boldsymbol{\pi}(r\text{-}1), j = \boldsymbol{\pi}(r), \boldsymbol{\pi}(r+1),\ldots, \boldsymbol{\pi}(k),\ldots,\boldsymbol{\pi}(n))$ преобразовать в порядок $\boldsymbol{\pi}' =(\boldsymbol{\pi}(1), \boldsymbol{\pi}(2),\ldots, \boldsymbol{\pi}(r\text{-}1), \boldsymbol{\pi}(r+1),\ldots, \boldsymbol{\pi}(k),\ldots, \boldsymbol{\pi}(n), j)$.
6) Отметить работу j. Перейти к 3).

В случае, если штрафная функция имеет вид: задача является универсальной переборной.

$$\psi_j = \begin{cases} 0, & \text{если } t_j \le d_j \\ c_j, & \text{если } t_j > d_j \end{cases}$$

Задачи о нескольких исполнителях

Имеется два идентичных исполнителя и n работ с длительностями $p_1, p_2, \ldots, p_n$
Нет никаких ограничений ни на разбрасывание, ни на порядок. Необходимо минимизировать время, когда закончит работу последний из двух исполнителей.

Для решения необходимо разбросать работы так, чтобы суммы длительностей работ, доставшихся каждому исполнителю были как можно более одинаковыми. (Идентична «Задаче о камнях», которая решается только перебором)

Задача С.Джонсона о двух станках: На двух станках M_1 и M_2 нужно изготовить n деталей. Для изготовления j-й детали нужно проделать 2 операции – сначала первую на станке M_1 с длительностью a_j, а потом(возможно после перерыва) – вторую, на станке M_2 с длительностью b_j. Требуется найти такие порядки выполнения операций на обоих станках, чтобы как можно раньше изготовить все детали.
Алгоритм (Джонсон) решения этой задачи опирается на том свойстве, что
$f(T(\boldsymbol{\pi}_\alpha, \boldsymbol{\pi}_\beta)) \geq f(T(\boldsymbol{\pi}_\alpha, \boldsymbol{\pi}_\alpha))$

Алгоритм (Джонсон) Нахождения оптимального расписания в двухоперационной задаче Джонсона:

1) Выбрать среди «невычеркнутых» входных данных: $a_1, a_2,\ldots,a_n; b_1, b_2,\ldots, b_n$ минимальное число;
2) Если минимальное число принадлежит массиву a (есть такое число a_i), то i-ю операцию назначить первой среди «невычеркнутых», а в противном случае – последней среди «невычеркнутых». «Вычеркнуть» соответствующие a_i и b_i из массива входных данных;
3) Если все a и b вычеркнуты, то КОНЕЦ, иначе вернуться к п.1.

Решить задачу со следующими входными данными:

i	1	2	3	4	5
a_i	4	4	30	6	2
b_i	5	1	4	30	3