Scheduling Theory and Applications

Alexander Lazarev

Lomonosov Moscow State University

National Research University Higher School of Economics

Moscow Institute of Physics and Technology (State University)

V.A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences (ICS RAS)

jobmath@mail.ru

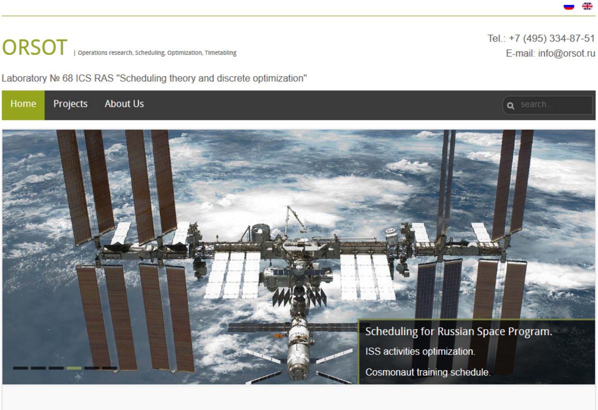
www.orsot.ru

Outline I

Outline II

Outline III

orsot.ru



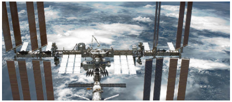
Laboratory's site

Laboratory №68 "Scheduling theory and Discrete Optimization"

Laboratory №68 of Scheduling Theory and Discrete Optimization was founded in 2009 at Institute of Control Sciences. Head of the laboratory is professor Alexander Lazarev. Currently it is the only laboratory in Russia studying problems of Scheduling Theory.

Our site orsot.ru (Operation Research Scheduling Optimization Timetabling)

Projects

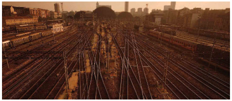


Optimization problems in astronautics
Scheduling for ISS (International Space Station) missions
Planning of cosmonauts training program

Managing railroad traffic
Managing railcar fleets
Operative management
Minimizing lateness and travel time

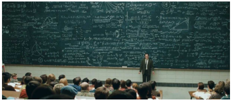
Strategical planning of manufacturing
Long-term and short-term planning
Minimizing production time
Uniform resource load

Projects



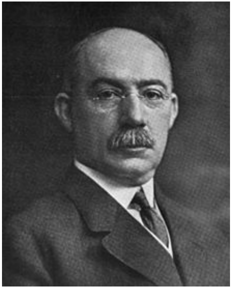
Transport logistics
Forming trains and routes

Optimizing assembly lines
Balancing and rebalancing assembly lines
Distributing operations



Composing study schedules
Program product on 1C platform

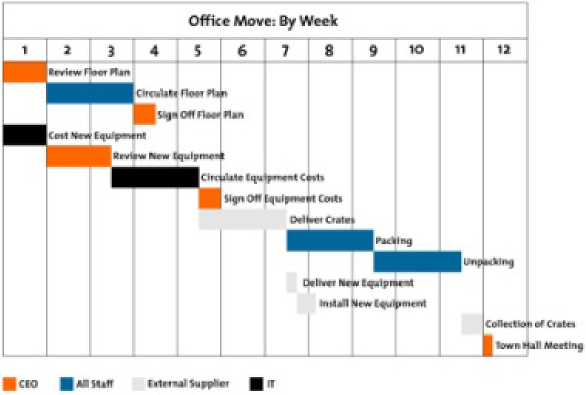
Gantt chart



Henry Laurence Gantt (1861-1919), American mechanical engineer and management consultant who is best known for his work in the development of scientific management. In the 1903 he introduced a graphical method of project schedule representation known as the Gantt chart (Gantt diagram).

"A graphical daily balance in manufacture"(1903)
"Organizing for Work"(1919)

Gantt chart



An example of Gantt chart

Scheduling theory term

Richard Ernest Bellman (1920–1984), American applied mathematician, famous for his work on dynamic programming and numerous important contributions in other fields of mathematics. In the 1954 he introduced the term "scheduling theory.

"Mathematical Aspects of Scheduling Theory"(1955)

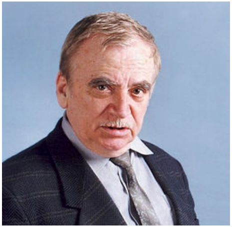
Pioneers of scheduling theory. First results.

J. R. Jackson. Scheduling a production to minimize maximum tardiness. Research Report 43, Management Science Research Project, University of California at Los Angeles, 1955

W. E. Smith. Various optimizers for single-stage production. Naval Research Logistic Quarterly, 3:59-66, 1956

S. M. Johnson. Optimal two-and-three-stage production schedules with set-up times included. Naval Research Logistics Quarterly, 1:61-68, 1954

Tanaev, V.S. and Shkurba, V.V. Vvedenie v teoriyu raspisanii (Introduction to the Scheduling Theory), Moscow: Nauka, 1975



J. R. Jackson.

Scheduling a production to minimize maximum tardiness. 1955.

J. R. Jackson. Scheduling a production to minimize maximum tardiness.

1 machine

J. R. Jackson. Scheduling a production to minimize maximum tardiness.

1 machine

n jobs, $N = \{1, 2, 3, \ldots, n\}$

J. R. Jackson. Scheduling a production to minimize maximum tardiness.

1 machine

n jobs, $N = \{1, 2, 3, \ldots, n\}$

r_j — release time

J. R. Jackson. Scheduling a production to minimize maximum tardiness.

1 machine

n jobs, $N = \{1, 2, 3, \ldots, n\}$

r_j — release time

p_j — processing time

J. R. Jackson. Scheduling a production to minimize maximum tardiness.

1 machine

n jobs, $N = \{1, 2, 3, \ldots, n\}$

r_j — release time

p_j — processing time

d_j — due date, D_j — deadline.
The due dates are allowed to be violated, but the deadlines are not.

J. R. Jackson. Scheduling a production to minimize maximum tardiness.

1 machine

n jobs, $N = \{1, 2, 3, \ldots, n\}$

r_j — release time

p_j — processing time

d_j — due date, D_j — deadline.
The due dates are allowed to be violated, but the deadlines are not.

Schedule π (permutation of jobs)

J. R. Jackson. Scheduling a production to minimize maximum tardiness.

Note: There are two most common ways of schedule representation:

— The schedule is represented by a permutation of jobs (in what order the jobs should be processed, one after one), for example, $\pi = (6, 3, 2, 1, \dots)$ means that firstly job 6 is processed, then job 3, then 2 and so on.
— The schedule is represented by a vector of job start times S_j, for example $\pi = (10, 0, 11, 5, 6, 4, \dots)$ means that job 1 starts at $t = 10$, job 2 starts at $t = 0$, 3 starts at $t = 11$ and so on.

Depending on the formulation of considered problem, one method or another may be more convenient to implement.

J. R. Jackson. Scheduling a production to minimize maximum tardiness.

1 machine

n jobs, $N = \{1, 2, 3, \ldots, n\}$

r_j — release time

p_j — processing time

d_j — due date, D_j — deadline.

The due dates are allowed to be violated, but the deadlines are not.

Schedule π (permutation of jobs)

J. R. Jackson. Scheduling a production to minimize maximum tardiness.

1 machine

n jobs, $N = \{1, 2, 3, \dots, n\}$

r_j — release time

p_j — processing time

d_j — due date, D_j — deadline.

The due dates are allowed to be violated, but the deadlines are not.

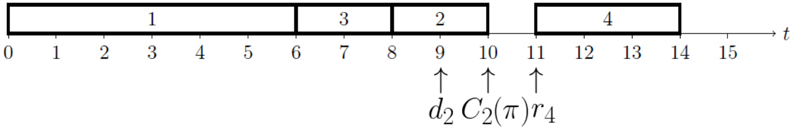
Schedule π (permutation of jobs)

$C_j(\pi)$ — completion time of job j

J. R. Jackson. Scheduling a production to minimize maximum tardiness.

1 machine
n jobs, $N = \{1, 2, 3, \dots, n\}$
r_j — release time
p_j — processing time
d_j — due date, D_j — deadline.
The due dates are allowed to be violated, but the deadlines are not.
Schedule π (permutation of jobs)
$C_j(\pi)$ — completion time of job j

schedule π

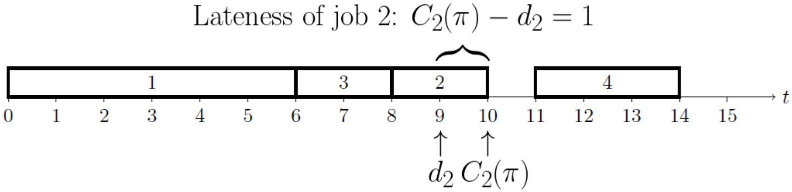


J. R. Jackson. Scheduling a production to minimize maximum tardiness.

Lateness of job j: $C_j(\pi) - d_j$

J. R. Jackson. Scheduling a production to minimize maximum tardiness.

Lateness of job j: $C_j(\pi) - d_j$



J. R. Jackson. Scheduling a production to minimize maximum tardiness.

Lateness of j-th job $C_j(\pi) - d_j$

The goal is to construct a schedule with minimal value of maximum lateness:

$$\min_{\pi} \max_{j \in N} \{ C_j(\pi) - d_j \}$$

Think, how you would solve this problem.

J. R. Jackson. Scheduling a production to minimize maximum tardiness.

Lateness of j-th job $C_j(\pi) - d_j$

The goal is to construct a schedule with minimal value of maximum lateness:

$$\min_{\pi} \max_{j \in N} \{ C_j(\pi) - d_j \}$$

Think, how you would solve this problem.

Jackson's result: if all release times are zero, $\forall j \in N \; r_j = 0$, then optimal schedule $\pi^* = (j_1, j_2, \ldots, j_n)$ consists of jobs that are sorted according to non-decrease of their due dates:

$$d_{j_1} \leq d_{j_2} \leq \cdots \leq d_{j_n}$$

Optimal schedule can be obtained by using sorting algorithm with $O(n \log n)$ operations

J. R. Jackson. Scheduling a production to minimize maximum tardiness.

Jackson's algorithm requires that all the jobs are accessible from the beginning ($\forall j \in N \; r_j = 0$).

J. R. Jackson. Scheduling a production to minimize maximum tardiness.

Jackson's algorithm requires that all the jobs are accessible from the beginning ($\forall j \in N \; r_j = 0$).

What if this requirement is not met ($\exists \, j \in N \; r_j \neq 0$)?

W. E. Smith.

Various optimizers for single-stage production. 1956

W. E. Smith. Various optimizers for single-stage production.

1 machine

W. E. Smith. Various optimizers for single-stage production.

1 machine

n jobs, $N = \{1, 2, 3, \ldots, n\}$

W. E. Smith. Various optimizers for single-stage production.

1 machine

n jobs, $N = \{1, 2, 3, \ldots, n\}$

$r_j = 0$, $\forall j \in N$ — all jobs are released at $t = 0$

W. E. Smith. Various optimizers for single-stage production.

1 machine

n jobs, $N = \{1, 2, 3, \ldots, n\}$

$r_j = 0$, $\forall j \in N$ — all jobs are released at $t = 0$

p_j — processing time

W. E. Smith. Various optimizers for single-stage production.

1 machine

n jobs, $N = \{1, 2, 3, \ldots, n\}$

$r_j = 0$, $\forall j \in N$ — all jobs are released at $t = 0$

p_j — processing time

Schedule π (permutation of jobs)

W. E. Smith. Various optimizers for single-stage production.

1 machine

n jobs, $N = \{1, 2, 3, \ldots, n\}$

$r_j = 0$, $\forall j \in N$ — all jobs are released at $t = 0$

p_j — processing time

Schedule π (permutation of jobs)

$C_j(\pi)$ — completion time

W. E. Smith. Various optimizers for single-stage production.

1 machine

n jobs, $N = \{1, 2, 3, \ldots, n\}$

$r_j = 0,\ \forall j \in N$ — all jobs are released at $t = 0$

p_j — processing time

Schedule π (permutation of jobs)

$C_j(\pi)$ — completion time

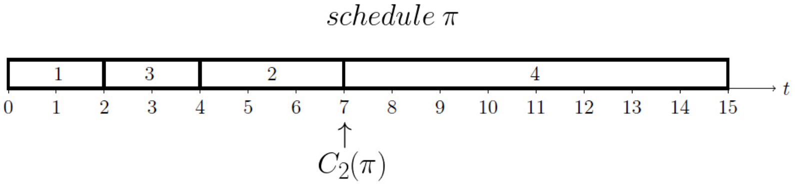
Objective function: minimum total completion time

$$\min_{\pi} \sum_{j \in N} C_j(\pi)$$

W. E. Smith. Various optimizers for single-stage production.

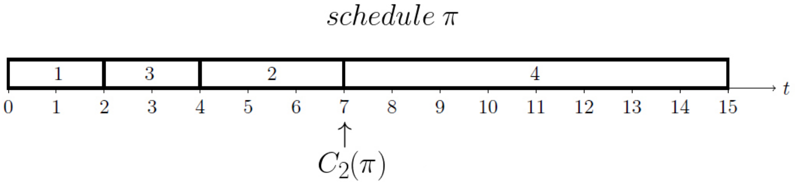
Smith's result: in optimal schedule $\pi^* = (j_1, j_2, \ldots, j_n)$ jobs are sorted according to non-decrease of their processing times:

$$p_{j_1} \leq p_{j_2} \leq \cdots \leq p_{j_n}$$



Smith's result: in optimal schedule $\pi^* = (j_1, j_2, \ldots, j_n)$ jobs are sorted according to non-decrease of their processing times:

$$p_{j_1} \leq p_{j_2} \leq \cdots \leq p_{j_n}$$



Optimal schedule can be obtained by using sorting algorithm with $O(n \log n)$ operations

In Smith's problem all jobs are released at $t = 0$ ($\forall j \in N\, r_j = 0$).

W. E. Smith. Various optimizers for single-stage production.

In Smith's problem all jobs are released at $t = 0$ ($\forall j \in N\, r_j = 0$).
What if this requirement is not met ($\exists\, j \in N\, r_j \neq 0$)?

S. M. Johnson.

Optimal two-and-three-stage production schedules with set-up times included. 1954

S. M. Johnson. Optimal two-and-three-stage production schedules with set-up times included.

2 machines

S. M. Johnson. Optimal two-and-three-stage production schedules with set-up times included.

2 machines

N_{12} jobs with processing order "Machine 1 $\rightarrow$ Machine 2

S. M. Johnson. Optimal two-and-three-stage production schedules with set-up times included.

2 machines

N_{12} jobs with processing order "Machine 1 $\rightarrow$ Machine 2

p_j^1, p_j^2 — processing times of job j on machines 1 ("set-up time") and 2 ("actual"processing time), respectively

S. M. Johnson. Optimal two-and-three-stage production schedules with set-up times included.

2 machines

N_{12} jobs with processing order "Machine 1 $\rightarrow$ Machine 2

p_j^1, p_j^2 — processing times of job j on machines 1 ("set-up time") and 2 ("actual"processing time), respectively

$C_j^1(\pi), C_j^2(\pi)$ — completion times on machines 1 and 2 respectively

S. M. Johnson. Optimal two-and-three-stage production schedules with set-up times included.

2 machines

N_{12} jobs with processing order "Machine 1 $\rightarrow$ Machine 2

p_j^1, p_j^2 — processing times of job j on machines 1 ("set-up time") and 2 ("actual"processing time), respectively

$C_j^1(\pi), C_j^2(\pi)$ — completion times on machines 1 and 2 respectively

Schedule π (permutation of jobs, each job should be processed on machine 1 first)

S. M. Johnson. Optimal two-and-three-stage production schedules with set-up times included.

2 machines

N_{12} jobs with processing order "Machine 1 $\rightarrow$ Machine 2

$p_j^1,\ p_j^2$ — processing times of job j on machines 1 ("set-up time") and 2 ("actual"processing time), respectively

$C_j^1(\pi), C_j^2(\pi)$ — completion times on machines 1 and 2 respectively

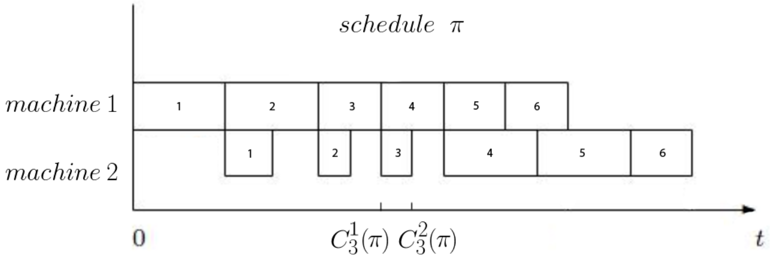
Schedule π (permutation of jobs, each job should be processed on machine 1 first)

Objective function: minimum total processing time (makespan)

$$\min_{\pi} \max_{j \in N} \{ C_j^2(\pi) \}$$

S. M. Johnson. Optimal two-and-three-stage production schedules with set-up times included.

Example of a feasible schedule:



S. M. Johnson. Optimal two-and-three-stage production schedules with set-up times included.

Constraints:

1. Each machine may process only one job at a time

S. M. Johnson. Optimal two-and-three-stage production schedules with set-up times included.

Constraints:

1. Each machine may process only one job at a time
2. Each job may be processed at machine 2 only after it was processed at machine 1, i.e. moment of processing completion of job j at machine 1 cannot exceed moment of processing initiation of the same job at machine 2

S. M. Johnson. Optimal two-and-three-stage production schedules with set-up times included.

Constraints:

1. Each machine may process only one job at a time
2. Each job may be processed at machine 2 only after it was processed at machine 1, i.e. moment of processing completion of job j at machine 1 cannot exceed moment of processing initiation of the same job at machine 2
3. Processing of any job cannot be interrupted: if processing of job i on machine j was initiated at the moment of time t , it should remain processing on the same machine until the moment of time $t + p_i^j$

Johnson's algorithm:

Algorithm 1. Input: set N_{12} of jobs. Output: permutation π.

Step 1 $\forall i \in N_{12}\ p_i := min\{p_i^1, p_i^2\}$
Sorting jobs according to increase of their processing duration

$$p_{i_1} \leq p_{i_2} \leq \cdots \leq p_{i_n}$$

Johnson's algorithm:

Algorithm 1. Input: set N_{12} of jobs. Output: permutation π.

Step 1 $\forall i \in N_{12}\ p_i := min\{p_i^1, p_i^2\}$
Sorting jobs according to increase of their processing duration

$$p_{i_1} \leq p_{i_2} \leq \cdots \leq p_{i_n}$$

Step 2 $\pi_1 := \emptyset\ \ \pi_2 := \emptyset$

Johnson's algorithm:

Algorithm 1. Input: set N_{12} of jobs. Output: permutation π.

Step 1 $\forall i \in N_{12}$ $p_i := min\{p_i^1, p_i^2\}$
Sorting jobs according to increase of their processing duration

$$p_{i_1} \leq p_{i_2} \leq \cdots \leq p_{i_n}$$

Step 2 $\pi_1 := \emptyset$ $\pi_2 := \emptyset$

Step 3 Let N be a set of jobs sorted according to Step 1.
If $N = \emptyset$, go to step 4.
Otherwise, let's denote the first element of N as i_1. If $p_{i_1} = p_{i_1}^1$, add it to π_1: $\pi_1 := \pi_1 \cup$ i_1, otherwise, if $p_{i_1} = p_{i_1}^2$, add it to π_2: $\pi_2 :=$ i_1 $\cup \pi_2$

Johnson's algorithm:

Algorithm 1. Input: set N_{12} of jobs. Output: permutation π.

Step 1 $\forall i \in N_{12}\ p_i := min\{p_i^1, p_i^2\}$
Sorting jobs according to increase of their processing duration

$$p_{i_1} \leq p_{i_2} \leq \cdots \leq p_{i_n}$$

Step 2 $\pi_1 := \emptyset\ \ \pi_2 := \emptyset$

Step 3 Let N be a set of jobs sorted according to Step 1.
If $N = \emptyset$, go to step 4.
Otherwise, let's denote the first element of N as i_1. If $p_{i_1} = p_{i_1}^1$, add it to π_1: $\pi_1 := \pi_1 \cup\ i_1$, otherwise, if $p_{i_1} = p_{i_1}^2$, add it to π_2: $\pi_2 := i_1\ \cup \pi_2$

Step 4 $\pi := \pi_1 \cup \pi_2$
End.

Johnson's algorithm:

Algorithm 2. Input: permutation of jobs π. Output: feasible schedule.

Step 1 Moment of processing initiation of 1st job on machine 1 is 0. Moment of processing initiation of each subsequent job on machine 1 equals to moment of processing completion of previous job on machine 1.

Johnson's algorithm:

Algorithm 2. Input: permutation of jobs π. Output: feasible schedule.

Step 1 Moment of processing initiation of 1st job on machine 1 is 0. Moment of processing initiation of each subsequent job on machine 1 equals to moment of processing completion of previous job on machine 1.

Step 2 Moment of processing initiation of 1st job on machine 2 matches its moment of processing completion on machine 1. Moment of processing initiation of each subsequent job on machine 2 equals to maximum of two moments: moment of its processing completion on machine 1 and moment of processing completion of previous job on machine 2.

Algorithm 2. Input: permutation of jobs π. Output: feasible schedule.

Step 1 Moment of processing initiation of 1st job on machine 1 is 0. Moment of processing initiation of each subsequent job on machine 1 equals to moment of processing completion of previous job on machine 1.

Step 2 Moment of processing initiation of 1st job on machine 2 matches its moment of processing completion on machine 1. Moment of processing initiation of each subsequent job on machine 2 equals to maximum of two moments: moment of its processing completion on machine 1 and moment of processing completion of previous job on machine 2.

End.

Thus, overall computational complexity of Johnson's algorithm is limited by computational complexity of sorting algorithm implemented in Algorithm 1 at Step 1, i. e. $O(n \log n)$ in case of "quick-sort

Johnson's algorithm:

Algorithm 2. Input: permutation of jobs π. Output: feasible schedule. Here, schedule is described by an array of numbers: for each job j, processing start times S_j^1 and S_j^2 on machines 1 and 2 are assigned: $C_j^i = S_j^i + p_j^i,\ i = 1, 2,$

$$\pi = (j_1, j_2, , \dots, j_n)$$

Johnson's algorithm:

Algorithm 2. Input: permutation of jobs π. Output: feasible schedule. Here, schedule is described by an array of numbers: for each job j, processing start times S_j^1 and S_j^2 on machines 1 and 2 are assigned: $C_j^i = S_j^i + p_j^i,\ i = 1, 2,$

$$\pi = (j_1, j_2, , \dots, j_n)$$

Johnson's algorithm:

Algorithm 2. Input: permutation of jobs π. Output: feasible schedule. Here, schedule is described by an array of numbers: for each job j, processing start times S_j^1 and S_j^2 on machines 1 and 2 are assigned: $C_j^i = S_j^i + p_j^i,\ i = 1, 2,$

$$\pi = (j_1, j_2, , \ldots, j_n)$$

Exercise 1.

j	1	2	3	4	5	6	7
p_j^1	5	7	4	3	5	7	6
p_j^2	6	5	6	7	4	6	8

$\pi = (\ ,\ ,\ ,\ ,\ ,\ ,\ )$
$\pi_1 = ()$
$\pi_2 = ()$

Final schedule π will be composed of two parts: $\pi = \pi_1 \cup \pi_2$. π_1 contains jobs that should be performed on machine 1 first. After all the jobs from π_1 have been processed on machine 1, jobs from π_2 may start processing on that machine.

Exercise 1.

j	1	2	3	4	5	6	7
p_j^1	5	7	4	3	5	7	6
p_j^2	6	5	6	7	4	6	8

$\pi = (4\ ,\ \ ,\ \ ,\ \ ,\ \ ,\ \ ,\ \ )$
$\pi_1 = (4)$
$\pi_2 = ()$
Exclude job 4 from the list of pending jobs

Exercise 1.

j	1	2	3	5	6	7
p_j^1	5	7	4	5	7	6
p_j^2	6	5	6	4	6	8

$\pi = (4\ ,\ \ ,\ \ ,\ \ ,\ \ ,\ \ ,\ \ )$
$\pi_1 = (4)$
$\pi_2 = ()$

Exercise 1.

j	1	2	3	5	6	7
p_j^1	5	7	4	5	7	6
p_j^2	6	5	6	4	6	8

$\pi = (4\ ,\ \ ,\ \ ,\ \ ,\ \ ,\ \ ,5)$
$\pi_1 = (4)$
$\pi_2 = (5)$
Exclude job 5 from the list of pending jobs

Exercise 1.

j	1	2	3	6	7
p_j^1	5	7	4	7	6
p_j^2	6	5	6	6	8

$\pi = (4\ , 3\ ,\ \ ,\ \ ,\ \ ,\ \ , 5)$
$\pi_1 = (4, 3)$
$\pi_2 = (5)$
Exclude job 3 from the list of pending jobs

Exercise 1.

j	1	2	6	7
p_j^1	5	7	7	6
p_j^2	6	5	6	8

$\pi = (4\ , 3\ ,\ \ ,\ \ ,\ \ ,\ \ , 5)$
$\pi_1 = (4, 3)$
$\pi_2 = (5)$

Exercise 1.

j	1	2	6	7
p_j^1	5	7	7	6
p_j^2	6	5	6	8

$\pi = (4\ , 3\ ,\ \ ,\ \ ,\ \ , 2\ , 5)$
$\pi_1 = (4, 3)$
$\pi_2 = (2, 5)$
Exclude job 2 from the list of pending jobs

Exercise 1.

j	1	6	7
p_j^1	5	7	6
p_j^2	6	6	8

$\pi = (4\ ,3\ ,1\ ,\ \ ,\ \ ,2\ ,5)$
$\pi_1 = (4, 3, 1)$
$\pi_2 = (2, 5)$
Exclude job 1 from the list of pending jobs

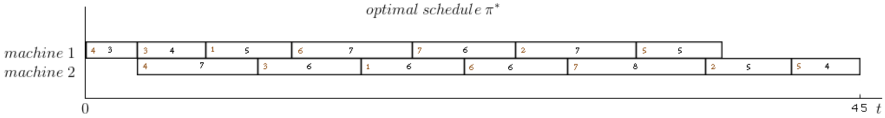
Exercise 1.

j	6	7
p_j^1	7	6
p_j^2	6	8

$\pi = (4, 3, 1, 6, 7, 2, 5)$ $\quad \pi_1 = (4, 3, 1, 6)$ $\quad \pi_2 = (7, 2, 5)$

$\pi = \pi_1 \bigcup \pi_2$

$O(n \log n)$



S. M. Johnson. Optimal two-and-three-stage production schedules with set-up times included.

So far we have only considered the case in which all the jobs have processing order "Machine 1 $\rightarrow$ Machine 2"(further, we will denote it simply as "1 $\rightarrow$ 2").

S. M. Johnson. Optimal two-and-three-stage production schedules with set-up times included.

So far we have only considered the case in which all the jobs have processing order "Machine 1 $\to$ Machine 2"(further, we will denote it simply as "$1 \to 2$").

Let us consider a bit more complicated problem. What if there are not only jobs with a processing order "$1 \to 2$ but also with processing orders "$2 \to 1$ "1"and "2"? (In the latter two cases, the jobs should only be processed on their respective machines).

S. M. Johnson. Optimal two-and-three-stage production schedules with set-up times included.

So far we have only considered the case in which all the jobs have processing order "Machine 1 $\rightarrow$ Machine 2"(further, we will denote it simply as "1 $\rightarrow$ 2").

Let us consider a bit more complicated problem. What if there are not only jobs with a processing order "1 $\rightarrow$ 2 but also with processing orders "2 $\rightarrow$ 1 "1"and "2"? (In the latter two cases, the jobs should only be processed on their respective machines).

How could we apply Johnson's algorithm to this problem?

S. M. Johnson. Optimal two-and-three-stage production schedules with set-up times included.

2 machines

S. M. Johnson. Optimal two-and-three-stage production schedules with set-up times included.

2 machines

N_1 jobs with processing order "1"

N_2 jobs with processing order "2"

N_{12} jobs with processing order "$1 \to 2$"

N_{21} jobs with processing order "$2 \to 1$"

S. M. Johnson. Optimal two-and-three-stage production schedules with set-up times included.

2 machines

N_1 jobs with processing order "1"

N_2 jobs with processing order "2"

N_{12} jobs with processing order "$1 \to 2$"

N_{21} jobs with processing order "$2 \to 1$"

$p_j^1,\ p_j^2$ — processing times of job j on machines 1 and 2 respectively

(consider $\forall j \in N_1\ p_j^2 = 0, \quad \forall j \in N_2\ p_j^1 = 0$)

S. M. Johnson. Optimal two-and-three-stage production schedules with set-up times included.

2 machines
N_1 jobs with processing order "1"
N_2 jobs with processing order "2"
N_{12} jobs with processing order "$1 \to 2$"
N_{21} jobs with processing order "$2 \to 1$"
$p_j^1,\ p_j^2$ — processing times of job j on machines 1 and 2 respectively
(consider $\forall j \in N_1\ p_j^2 = 0, \quad \forall j \in N_2\ p_j^1 = 0$)
$C_j^1(\pi), C_j^2(\pi)$ — completion times on machines 1 and 2 respectively

S. M. Johnson. Optimal two-and-three-stage production schedules with set-up times included.

2 machines

N_1 jobs with processing order "1"

N_2 jobs with processing order "2"

N_{12} jobs with processing order "$1 \to 2$"

N_{21} jobs with processing order "$2 \to 1$"

$p_j^1,\ p_j^2$ — processing times of job j on machines 1 and 2 respectively

(consider $\forall j \in N_1\ p_j^2 = 0, \quad \forall j \in N_2\ p_j^1 = 0$)

$C_j^1(\pi), C_j^2(\pi)$ — completion times on machines 1 and 2 respectively

Schedule $\pi = (\pi^1, \pi^2)$ (<u>two</u> schedules, for machines 1 and 2 respectively)

S. M. Johnson. Optimal two-and-three-stage production schedules with set-up times included.

2 machines

N_1 jobs with processing order "1"

N_2 jobs with processing order "2"

N_{12} jobs with processing order "$1 \to 2$"

N_{21} jobs with processing order "$2 \to 1$"

$p_j^1,\ p_j^2$ — processing times of job j on machines 1 and 2 respectively (consider $\forall j \in N_1\ p_j^2 = 0, \quad \forall j \in N_2\ p_j^1 = 0$)

$C_j^1(\pi), C_j^2(\pi)$ — completion times on machines 1 and 2 respectively

Schedule $\pi = (\pi^1, \pi^2)$ (<u>two</u> schedules, for machines 1 and 2 respectively)

Objective function: minimum total processing duration (makespan)

$$\min_{\pi} \max_{i \in \{1,2\}, j \in N} \{C_j^i(\pi)\},\ N = N_1 \cup_2 \cup N_{12} \cup N_{21}$$

S. M. Johnson. Optimal two-and-three-stage production schedules with set-up times included.

Without going into any deep detail on this problem, let us formulate the following theorem:

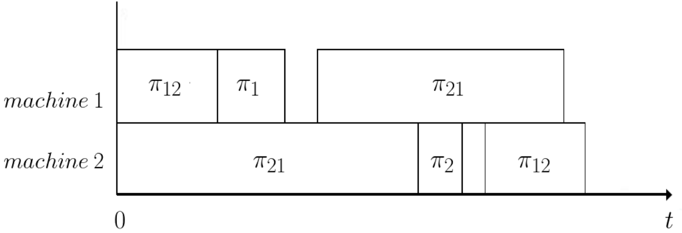
S. M. Johnson. Optimal two-and-three-stage production schedules with set-up times included.

Without going into any deep detail on this problem, let us formulate the following theorem:

Theorem. Let π_{12} be a permutation of jobs obtained by applying Algorithm 1 to set of jobs N_{12}, let π_{21} be a permutation of jobs obtained by applying Algorithm 1 to set of jobs N_{21} (by swapping the machines), and let π_1 and π_2 be two arbitrary permutations of jobs from sets N_1 and N_2. Then the optimal solution to of this problem would consist of two sequences of jobs: $\pi^1 = (\pi_{12}, \pi_1, \pi_{21})$ for machine 1 and $\pi^2 = (\pi_{21}, \pi_2, \pi_{12})$ for machine 2. Computational complexity of this algorithm is $O(n \log n)$, where n is total number of jobs.

S. M. Johnson. Optimal two-and-three-stage production schedules with set-up times included.

Schedule $\pi^1 = (\pi_{12}, \pi_1, \pi_{21})$ for machine 1
Schedule $\pi^2 = (\pi_{21}, \pi_2, \pi_{12})$ for machine 2



S. M. Johnson. Optimal two-and-three-stage production schedules with set-up times included.

Suppose that there are more than 2 machines, for example, 3 machines, and some jobs have processing orders such as "$1 \to 2 \to 3$ "$2 \to 3 \to 1$"and so on.

S. M. Johnson. Optimal two-and-three-stage production schedules with set-up times included.

Suppose that there are more than 2 machines, for example, 3 machines, and some jobs have processing orders such as "$1 \to 2 \to 3$ "$2 \to 3 \to 1$"and so on.

Is it possible to use Johnson's algorithm in that case?

Computational complexity of Jackson's, Smith's and Johnon's problems

The following problems:

— Smith's problem with non-zero release times ($\exists\, j \in N\ r_j \neq 0$)
— Jackson's problem with non-zero release times ($\exists\, j \in N\ r_j \neq 0$)
— Johnson's problem with more than 2 machines

are known to be at least NP-hard.

Meaning of the objective function in a problem

As we have discussed before, Jackson's, Smith's and Johnson's problems have objective functions L_{max}, $\sum C_j$ and C_{max}, correspondingly.

Meaning of the objective function in a problem

As we have discussed before, Jackson's, Smith's and Johnson's problems have objective functions L_{max}, $\sum C_j$ and C_{max}, correspondingly.

Now, take a moment and try to answer the following question: *What sense do these objective functions make in real life?*

Problem of two production lines

Problem of two production lines

2 production lines that have n workplaces denoted as $S_{11}, \dots, S_{1n}$ and $S_{21}, \dots, S_{2n}$, $i \in \{1, 2\}$, $j \in \{1. \dots, n\}$

Problem of two production lines

2 production lines that have n workplaces denoted as $S_{11}, \ldots, S_{1n}$ and $S_{21}, \ldots, S_{2n}$, $i \in \{1, 2\}$, $j \in \{1. \ldots, n\}$

1 job (product) that should pass all the n workplaces (stages of production) according to their order. Between stages of production the product may be transferred to another production line, i.e. different stages may be processed at different production lines.

Problem of two production lines

2 production lines that have n workplaces denoted as $S_{11}, \ldots, S_{1n}$ and $S_{21}, \ldots, S_{2n}$, $i \in \{1, 2\}$, $j \in \{1. \ldots, n\}$

1 job (product) that should pass all the n workplaces (stages of production) according to their order. Between stages of production the product may be transferred to another production line, i.e. different stages may be processed at different production lines.

a_{ij} — processing time at workplace S_{ij}

Problem of two production lines

2 production lines that have n workplaces denoted as $S_{11}, \ldots, S_{1n}$ and $S_{21}, \ldots, S_{2n}$, $i \in \{1, 2\}$, $j \in \{1. \ldots, n\}$

1 job (product) that should pass all the n workplaces (stages of production) according to their order. Between stages of production the product may be transferred to another production line, i.e. different stages may be processed at different production lines.

a_{ij} — processing time at workplace S_{ij}

t_{ij} — transfer time from workplace S_{ij} at production line i to $(j+1)$th workplace at the other production line; $j \in \{1, \ldots, n-1\}$. Transfer times between adjacent workplaces on the same production line are equal to 0.

Problem of two production lines

2 production lines that have n workplaces denoted as $S_{11}, \ldots, S_{1n}$ and $S_{21}, \ldots, S_{2n}$, $i \in \{1, 2\}$, $j \in \{1. \ldots, n\}$

1 job (product) that should pass all the n workplaces (stages of production) according to their order. Between stages of production the product may be transferred to another production line, i.e. different stages may be processed at different production lines.

a_{ij} — processing time at workplace S_{ij}

t_{ij} — transfer time from workplace S_{ij} at production line i to $(j+1)$th workplace at the other production line; $j \in \{1, \ldots, n-1\}$. Transfer times between adjacent workplaces on the same production line are equal to 0.

Problem of two production lines

2 production lines that have n workplaces denoted as $S_{11}, \dots, S_{1n}$ and $S_{21}, \dots, S_{2n}$, $i \in \{1, 2\}$, $j \in \{1. \dots, n\}$

1 job (product) that should pass all the n workplaces (stages of production) according to their order. Between stages of production the product may be transferred to another production line, i.e. different stages may be processed at different production lines.

a_{ij} — processing time at workplace S_{ij}

t_{ij} — transfer time from workplace S_{ij} at production line i to $(j+1)$th workplace at the other production line; $j \in \{1, \dots, n-1\}$. Transfer times between adjacent workplaces on the same production line are equal to 0.

Schedule π (each stage of the job is assigned to a workplace at the corresponding production line)

Problem of two production lines

$C_j(\pi)$ — completion time of stage j according to schedule π

Problem of two production lines

$C_j(\pi)$ — completion time of stage j according to schedule π

Objective function: minimum total processing duration (makespan):

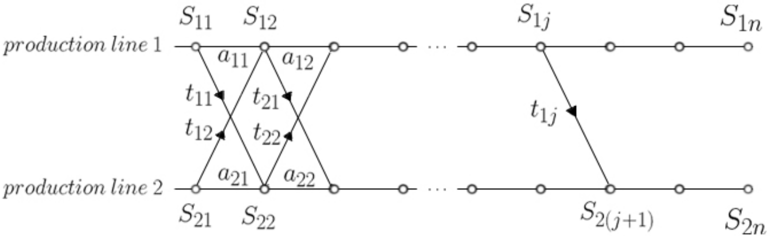
$$\min_{\pi}\{C_n(\pi)\}$$

Problem of two production lines

$C_j(\pi)$ — completion time of stage j according to schedule π

Objective function: minimum total processing duration (makespan):

$$\min_{\pi}\{C_n(\pi)\}$$



Solution. Dynamic programming.

Let us suppose that the job is now at the stage j at production line 1, i.e. the product is at workplace S_{1j}. Let us also suppose that the current schedule π is optimal.

In order to proceed to stage j, the job must have first gone through stage $j-1$, which means that the product came to workplace S_{1j} either from workplace $S_{1(j-1)}$ or $S_{2(j-1)}$. Suppose it came from workplace $S_{1(j-1)}$. According to our supposition that current schedule is optimal (which means that product got to workplace S_{1j} in the fastest possible way), the product must have gotten to workplace $S_{1(j-1)}$ in the fastest possible way, too. This means that the optimal solution of the problem for the first j workplaces includes optimal solution of the problem for the first $j-1$ workplaces. This property of the solution is called optimal substructure.

Solution. Dynamic programming.

Recursive algorithm: According to the optimal substructure of the problem, let us calculate consequently C_j^i — the least moments of time in which the product could have gone through stage j, being at the moment of completion of this stage at production line i.

$$
\begin{aligned}
&C_1^1 := a_{11}\\
&C_1^2 := a_{21}\\
&\mathit{for}\ j := 2\ \mathit{to}\ n\ \mathit{do}\\
&\mathit{begin}\\
&\quad C_j^1 := \min\{C_{j-1}^1, C_{j-1}^2 + t_{2(j-1)}\} + a_{1j}\\
&\quad C_j^2 := \min\{C_{j-1}^2, C_{j-1}^1 + t_{1(j-1)}\} + a_{2j}\\
&\mathit{end}
\end{aligned}
$$

These equations are the simplest case of Bellmann equations.

Solution. Dynamic programming.

Total processing duration is $C_n := min\{C_n^1, C_n^2\}$ i.e. it doesn't matter at which production line the product finished processing.
Recovering the schedule itself is a fairly easy task: we just have to "remember"from which workplace the product came to the current workplace.
Total computational complexity of this algorithm is $O(n)$ operations.

Elements of computational comlexity theory. Classes P and NP.

S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of 3rd Annual ACM Symposium on Theory of Computing*, pg. 151-158. ACM-Press 1971.

Computational complexity

Suppose we are examining some instance of a recognition problem.

Computational complexity

Suppose we are examining some instance of a recognition problem.

Let us denote as n a numerical characteristic of input data that affects computational complexity of this problem in the most significant way (usually it is either the amount of input data itself, or dimensionality of the problem — the number of variables, equations and inequalities that define an instance of the problem).

Computational complexity

Suppose we are examining some instance of a recognition problem.

Let us denote as n a numerical characteristic of input data that affects computational complexity of this problem in the most significant way (usually it is either the amount of input data itself, or dimensionality of the problem — the number of variables, equations and inequalities that define an instance of the problem).

Suppose we also know some sort of algorithm that can be used to solve this problem within a finite time period.

Computational complexity

• If computational complexity of the algorithm that solves the problem is $O(n^k)$ operations, where k is some constant number independent from n, then this problem is called solvable in polynomial time. Algorithms to the 4 problems mentioned before (Jackson's, Smith's, Johnson's problems and the problem of two production lines) are polynomial.

Computational complexity

• If computational complexity of the algorithm that solves the problem is $O(n^k)$ operations, where k is some constant number independent from n, then this problem is called solvable in polynomial time. Algorithms to the 4 problems mentioned before (Jackson's, Smith's, Johnson's problems and the problem of two production lines) are polynomial.

• All problems that are solvable within polynomial time formulate a class of problems denoted as P. Algorithms with corresponding computational complexity are called *polynomial*.

Computational complexity

• If computational complexity of the algorithm that solves the problem is $O(n^k)$ operations, where k is some constant number independent from n, then this problem is called solvable in polynomial time. Algorithms to the 4 problems mentioned before (Jackson's, Smith's, Johnson's problems and the problem of two production lines) are polynomial.

• All problems that are solvable within polynomial time formulate a class of problems denoted as P. Algorithms with corresponding computational complexity are called *polynomial*.

• If complexity of the algorithm depends on the values of numerical parameters of an example, for example, $O(nA)$, then this algorithm is called *pseudo-polynomial*.

Computational complexity

- If computational complexity of the algorithm that solves the problem is $O(n^k)$ operations, where k is some constant number independent from n, then this problem is called solvable in polynomial time. Algorithms to the 4 problems mentioned before (Jackson's, Smith's, Johnson's problems and the problem of two production lines) are polynomial.

- All problems that are solvable within polynomial time formulate a class of problems denoted as P. Algorithms with corresponding computational complexity are called *polynomial*.

- If complexity of the algorithm depends on the values of numerical parameters of an example, for example, $O(nA)$, then this algorithm is called *pseudo-polynomial*.

- If complexity of the algorithm has the form of $O(n^x y^n)$, where x and y are some constants, then this algorithm is called *exponential*.

Class NP

Suppose that we have a computer that includes a special "guessing"component (oracle). The oracle, given correct input data (the solution exists), provides some (possibly correct) output data. The output data provided by oracle needs to be verified, i. e. we should construct an algorithm that checks if the output data contains a correct solution that is in accordance with provided input data.

Class NP

Class NP includes all the problems that, for each instance that has a solution, may be guessed by an oracle, and the answer provided by oracle is such that:

Class NP

Class NP includes all the problems that, for each instance that has a solution, may be guessed by an oracle, and the answer provided by oracle is such that:

- the amount of data in solution provided by oracle is limited polynomially

Class NP

Class NP includes all the problems that, for each instance that has a solution, may be guessed by an oracle, and the answer provided by oracle is such that:

- the amount of data in solution provided by oracle is limited polynomially
- the solution provided by oracle could be verified in polynomial time.

Reduction of one problem to another

It is said that problem A can be reduced to problem B in polynomial time ($A \propto B$), if a modification algorithm exists, such that this algorithm follows two next conditions:

Reduction of one problem to another

It is said that problem A can be reduced to problem B in polynomial time ($A \propto B$), if a modification algorithm exists, such that this algorithm follows two next conditions:

• The algorithm transforms any given instance I_A of problem A into a corresponding instance I_B of problem B in polynomial time

Reduction of one problem to another

It is said that problem A can be reduced to problem B in polynomial time ($A \propto B$), if a modification algorithm exists, such that this algorithm follows two next conditions:

• The algorithm transforms any given instance I_A of problem A into a corresponding instance I_B of problem B in polynomial time

• The answer to received instance I_B of problem B is "**YES**" if and only if the answer to the corresponding instance I_A of problem A is "**YES**", too. (or, less strictly, the solutions of corresponding instances I_A, I_B of problems A, B always match)

Problem classification in scheduling theory

Problem classification in scheduling theory

In scheduling theory, problems are classified according to:

- Type of solution
- Type of objective function
- Way input data is provided
- Subfields of Scheduling Theory

Problem classification in scheduling theory

Problem classification according to the type of solution:

- Arrangement problems
- Matching problems
- Distribution problems

Problem classification in scheduling theory

Problem classification according to the type of objective function:

- Problems with summary optimization criteria
- Problems with min-max optimization criteria
- Multicriterial optimization problems
- Problem on constructing a feasible schedule

Problem classification in scheduling theory

Problem classification according to the way input data is provided:

- Deterministic problems (**offline**)
- Dynamic problems (**online**)

Problem classification in scheduling theory

Problem classification according to subfields of Scheduling Theory:

- Project scheduling (PS)
- Machine scheduling (MS)
- Timetabling
- Shop-floor scheduling
- Transport scheduling and vehicle routing
- Sports scheduling

Denotations in Scheduling Theory

In Scheduling Theory, tasks are referred to as *requests* or *jobs*. Parameters of requests:

- r_j — release time
- p_j — processing time
- d_j — due date (may be violated, but a penalty is issued)
- D_j — deadline (should never be violated)
- w_j — job weight

Denotations in Scheduling Theory

Additional denotations:

- *pmtn* — preemptive scheduling is allowed
- *prec* — precedence relations between the jobs are defined (also: *tree*, $out - tree$, $in - tree$, *chain*)
- *batch* — the batching problem is considered (jobs are grouped into batches)

Denotations in Scheduling Theory

Objective functions:

- C_j — completion time
- $L_j = C_j - d_j$ — lateness
- $T_j = \max\{0, C_j - d_j\}$ — tardiness
- $E_j = \max\{0, d_j - C_j\}$ — earliness
- U_j — equals 1 if job j is late $(C_j > d_j)$ and 0 in the opposite case

If request weights w_j are provided, all of the previous objective functions are called *weighed*, and are multiplied by the value of request weight (ex., weighed tardiness $w_j T_j$ is calculated as $w_j \max\{0, C_j - d_j\}$)

Denotations in Scheduling Theory

Optimization criteria:

1. min-max criteria
   - $C_{max} \to min$ — minimizing maximum completion time (makespan), $C_{max} = \max_{j \in N} C_j$. These problems are also called *performance problems*.
   - $L_{max} \to min$ — minimizing maximum lateness $L_{max} = \max_{j \in N} L_j$
2. summary criteria
   - $\sum_{j \in N} C_j \to min$ — minimizing total completion time
   - $\sum_{j \in N} T_j \to min$ — minimizing total tardiness
   - $\sum_{j \in N} U_j \to min$ — minimizing total number of late jobs

Also, problems of maximizing these objective functions are considered (ex., $\sum_{j \in N} T_j \to max$).

Project scheduling

Resource-Constrained Project Scheduling Problem (RCPSP)

- Set of n requests $N = \{1, \ldots, n\}$
- k renewable resources $K = 1, \ldots, Q_k$
- p_i — processing time of request i, $\forall i \in N$.
- During processing of request i amount $q_{ik} \leq Q_k$ of resource k is used, $k = 1, \ldots, n$.
- Some requests are bound by *precedence relations*:$i \to j$ means request j cannot start processing before request i has finished processing, $i, j \in N$.

Resource-Constrained Project Scheduling Problem (RCPSP)

The goal is to find processing start times S_i for all requests $i \in N$ so that minimum makespan C_{max} is achieved:

$$C_{max} = \max_{i \in N}\{C_i\},\ C_i = S_i + p_i,\ C_{max} \to min$$

Obtained schedule should comply to the following conditions:

- Resource constraints are not violated:
  $\forall\, t \in [0, C_{max}),\ \forall\, k = 1, \ldots, K\ \ \sum_{i=1}^{n} q_{ik}\varphi_i(t) \le Q_k$
- Precedence relations are not violated:
  $\forall\, i, j \in N$: if $i \to j$, then $S_i + p_i \le S_j$

It is necessary to notice that RCPSP is not the only problem in project scheduling, though it is the main one. For example, some resources can be non-renewable, such as money, fuel, oils and so on.

Machine scheduling

Machine scheduling

In Project Scheduling, processing of each request requires participation of several *processors* (renewable resources could be viewed as equipment).

In Machine scheduling, usually each request is processed by *only one processor at a time*.

Processors can also be referred to as *machines* or *devices*. If not specified otherwise, machines are considered equivalent.

• *Single-machine problems*: only one request can be processed at a time.

• *Parallel machines' problems*: each request can be processed by any of the machines. Machines can be non-equivalent (processing time can vary). Precedence relations can be specified.

• *Shop scheduling*: m, machines $M_1, \ldots, M_m$. Each request $j \in N$ includes a number of *stages* ("operations") $O_1, \ldots, O_{n_j j}$. Precedence relations between operations can be specified. Each operation O_{ij} is assigned to a machine μ_{ij} that it should be processed on. For each request, only one operation can be processed at a time. Each machine can only process one operation at a time.

• *Job-shop*: Precedence relations between operations are $O_{1j} \to O_{2j} \to \cdots \to O_{n_j j}$. No precedence relations between requests. Number of operations may vary between requests.

- *Flow-shop ("Conveyor problem")*: Each request contains the same number of operations: $\forall j \in N \ n_j = m$. Same operations are assigned to the same machine: $\mu_{ij} = M_i$, $i = 1, \ldots, m$, $j = 1, \ldots, n$.
- *Open-shop*: same as Flow-shop, but no precedence relations between operations.
- *Other problems*: batching problems, multiprocessor problems, ...

Classification of problems in Machine scheduling

Classification of problems in Machine scheduling

Each problem is denoted as $\alpha|\beta|\gamma$, where

- α describes characteristics of the problem that are related to machines
- β describes constraints and conditions of processing of requests.
- γ describes objective function.

Classification of problems in Machine scheduling

α describes characteristics of the problem related to machines. Possible values of α:

- 1 — single machine
- Pm — parallel machines
- Qm — parallel machines (non-equivalent)
- Fm — Flow-shop problem
- Om — Open-shop problem
- Jm — Job-shop problem
- Other values: na, nd, ...

Classification of problems in Machine scheduling

β describes constraints and conditions of processing of requests. Possible contents of field β:

- r_j — release dates are specified
- d_j — due dates are specified
- D_j — deadlines are specified
- *prec* — precedence relations are specified
- *pmnt* — preemption is allowe
- *batch* — batching problem: groups of requests (*batches*) can be processed simultaneously.
- Other values: $p_j = p, \dots$

γ describes objective function (ex., C_{max}).

Classification of problems in Machine scheduling

Thus, record $F2|r_j|C_{max}$ denotes problem of minimizing makespan in Flow-shop system with two machines in case of non-simultaneous admission of requests. Other examples: $1|p_j = p,\ r_j|\sum w_j T_j$, $Pm|r_j,\ pmtn|\sum C_j$, ...

Thus, record $F2|r_j|C_{max}$ denotes problem of minimizing makespan in Flow-shop system with two machines in case of non-simultaneous admission of requests. Other examples: $1|p_j = p, r_j|\sum w_j T_j$, $Pm|r_j, pmtn|\sum C_j, \dots$

Let's review some of previously considered problems in terms of machine scheduling:

- $1|r_j|L_{max}$ (Jackson's problem with non-zero release times) is NP-hard in the strong sense
- $1|r_j|\sum C_j$ (Smith's problem with non-zero release times) is NP-hard
- $F3||C_{max}$ (Johnson's problem with more than 2 machines) is NP-hard in the strong sense

Minimizing maximum lateness

$1|r_j|L_{\max}$

Single machine, n jobs

r_j – release time;

$p_j > 0$ – processing time;

d_j – due date.

$j \in N = \{1, 2, \ldots, n\}$

Minimizing maximum lateness

$1|r_j|L_{\max}$

Single machine, n jobs
r_j – release time;
$p_j > 0$ – processing time;
d_j – due date.
$j \in N = \{1, 2, \ldots, n\}$

Preemptions of a job are not allowed. The machine can process at most one job at any time.

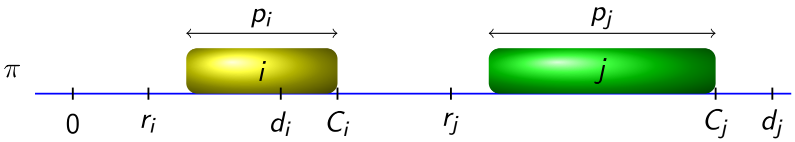
Minimizing maximum lateness

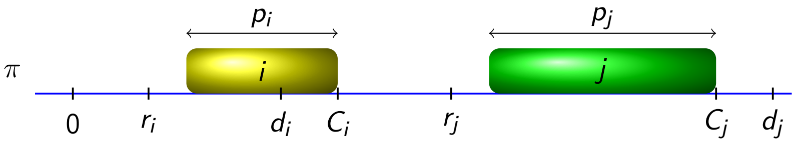
$1|r_j|L_{\max}$

Single machine, n jobs
r_j – release time;
$p_j > 0$ – processing time;
d_j – due date.
$j \in N = \{1, 2, \ldots, n\}$

Preemptions of a job are not allowed. The machine can process at most one job at any time.

Graham R.L., Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G. **1979**


p_i
p_j
π
i
j
0
r_i
d_i
C_i
r_j
C_j
d_j



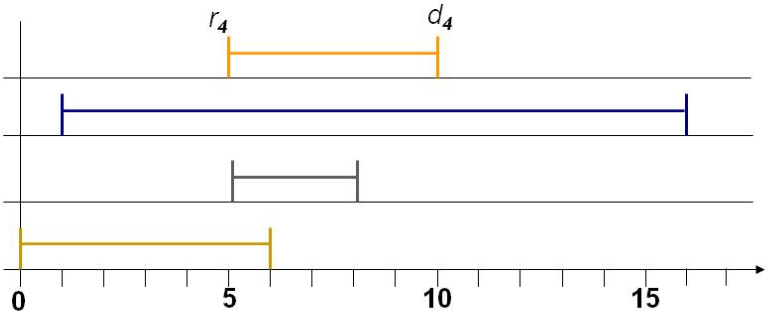
$$F(\pi) = \max_{j \in N}\{C_j - d_j\} \to \min_{\pi}$$

NP-hard in strong sense

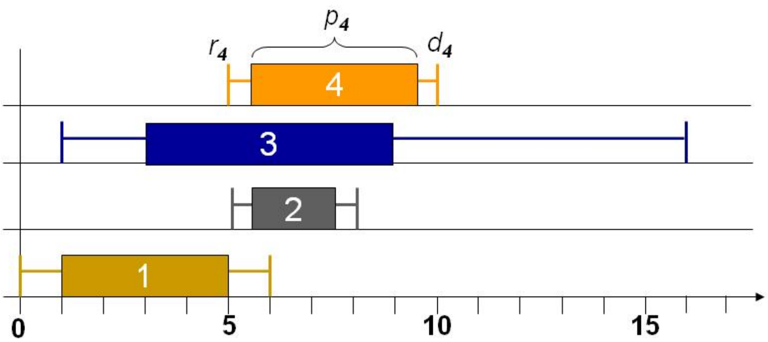
Lenstra J.K., Rinnooy Kan A.H.G., Brucker, P. **1977**

$1|r_j|L_{max}$

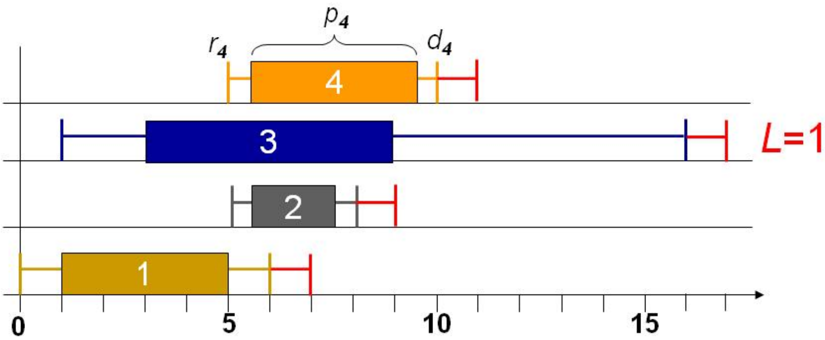
$1|r_j|L_{\max}$



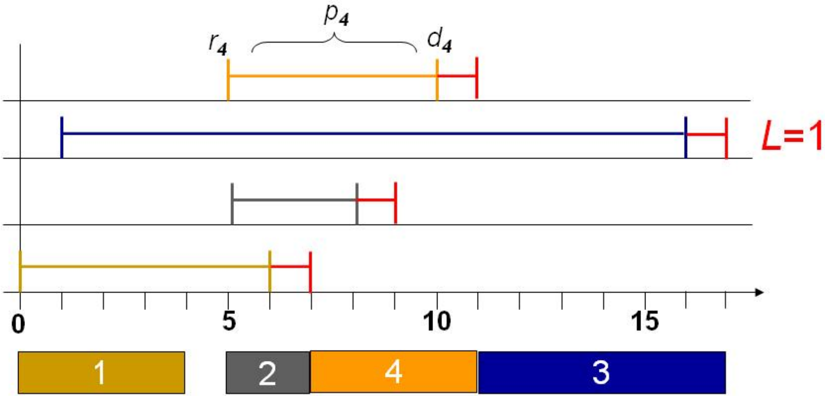
$1|r_j|L_{max}$



$1|r_j|L_{max}$



$1|r_j|L_{max}$



Solvable cases:

Solvable cases:

1) $r_j = 0, \forall\ j \in N.$ $O(n \log n)$
Jackson J.R. **1955**

1') $d_j = const, \forall\ j \in N.$ $O(n \log n)$

1") $p_j = const, \forall\ j \in N.$
Simons B. **1983**. $O(n \log n)$

Solvable cases:

1) $r_j = 0, \forall\ j \in N.$ $O(n \log n)$
Jackson J.R. **1955**

1') $d_j = const, \forall\ j \in N.$ $O(n \log n)$

1'') $p_j = const, \forall\ j \in N.$
Simons B. **1983**. $O(n \log n)$

2) $O(n^3 \log n)$

$$\begin{cases} d_1 \le d_2 \le \dots \le d_n; \\ d_1 - r_1 - p_1 \ge d_2 - r_2 - p_2 \ge \dots \ge d_n - r_n - p_n. \end{cases} \tag{1}$$

2') $d_j = r_j + p_j + const, \forall\ j \in N.$ $O(n^3 \log n)$

$\{1, P, Q, R\}|r_j|\{L_{\max}, C_{\max}\}$ $O(n^3 \log n)$

Lazarev A.A., Sadykov R.R., Sevastyanov S.V. **1988-2007**

Solvable cases:

Solvable cases:

3) $\max_{k \in N}\{d_k - r_k - p_k\} \leq d_j - r_j, \forall\, j \in N.$ $O(n^2 \log n)$

Hoogeveen J. A. **1996**

Solvable cases:

3) $\max\limits_{k \in N}\{d_k - r_k - p_k\} \le d_j - r_j, \forall\, j \in N.$ $O(n^2 \log n)$

Hoogeveen J. A. **1996**

4) *NP*-hard in ordinary sense $O(n^2 P + np_{\max} P)$

$$\begin{cases} d_1 \le d_2 \le \dots \le d_n; \\ r_1 \ge r_2 \ge \dots \ge r_n; \\ r_j, p_j, d_j \in \mathbb{Z}^+, \forall\, j \in N. \end{cases} \tag{2}$$

Lazarev A.A., Schulgina O.N. **1998**

$$P = r_{\max} + \sum_{j=1}^{n} p_j - r_{\min},\ r_{\max} = \max_{j \in N} r_j,\ r_{\min} = \min_{j \in N} r_j,\ p_{\max} = \max_{j \in N} p_j$$

Solvable cases:

Solvable cases:

5)

$$\begin{cases} d_1 \leq d_2 \leq \cdots \leq d_n; \\ d_1 - \alpha r_1 - \beta p_1 \geq d_2 - \alpha r_2 - \beta p_2 \geq \cdots \geq d_n - \alpha r_n - \beta p_n; \\ \alpha \in [1, \infty), \beta \in [0, 1]. \end{cases} \tag{3}$$

Solvable cases:

5)

$$\begin{cases} d_1 \le d_2 \le \dots \le d_n; \\ d_1 - \alpha r_1 - \beta p_1 \ge d_2 - \alpha r_2 - \beta p_2 \ge \dots \ge d_n - \alpha r_n - \beta p_n; \\ \alpha \in [1, \infty), \beta \in [0, 1]. \end{cases} \quad (3)$$

5')

$d_j = \alpha r_j + \beta p_j + const, \ \forall\ j \in N, \alpha \in [1, \infty), \beta \in [0, 1].$ **2009**

Solvable cases:

5)

$$\begin{cases} d_1 \le d_2 \le \cdots \le d_n; \\ d_1 - \alpha r_1 - \beta p_1 \ge d_2 - \alpha r_2 - \beta p_2 \ge \cdots \ge d_n - \alpha r_n - \beta p_n; \\ \alpha \in [1, \infty), \beta \in [0, 1]. \end{cases} \quad (3)$$

5')

$d_j = \alpha r_j + \beta p_j + const, \ \forall\ j \in N, \alpha \in [1, \infty), \beta \in [0, 1].$ **2009**

$O(n^3 \log n)$

Algorithm 1. $O(n \log n)$

Step 0) $\omega = \oslash$; $t = -\infty$;

Step 1) $f := f(N,t)$ and $s := s(N,t)$;

$$f(N,t) = \arg \min_{j \in N} \{d_j \mid r_j(t) = r(N,t)\},$$

$$s(N,t) = \arg \min_{j \in N \setminus \{f\}} \{d_j \mid r_j(t) = r(N \setminus \{f\}, t)\},$$

$$r_j(t) = \max\{r_j, t\}, r(N,t) = \min_{j \in N} \{r_j(t)\}.$$

Step 2) if $d_f \le d_s$ then

begin

$\omega := (\omega, f)$; $N := N \setminus \{f\}$, $t := r_f(t) + p_f$ and goto Step 1)

end

else RETURN.

Algorithm 2. $\alpha \in [1, \infty), \beta \in [0, 1]$ $O(n^2 \log n)$

$1 \mid d_i \leq d_j, d_i - \alpha r_i - \beta p_i \geq d_j - \alpha r_j - \beta p_j; L_{\max} \leq y \mid C_{\max}$

Step 0) $\theta := \omega(N, t)$; if $L_{\max}(\theta, t) > y$ then $\theta := \oslash$ and RETURN.

Step 1) $N := N \setminus \{\theta\}$; $t := C_{\max}(\theta)$;
$\omega^1 = (f, \omega(N \setminus \{f\}, r_f(t) + p_f); \omega^2 = (s, \omega(N \setminus \{s\}, r_s(t) + p_s)$;
if $L_{\max}(\omega^1, t) \leq y$ then $\theta := (\theta, \omega^1)$ and goto Step 1);

Step 2) if $L_{\max}(\omega^1, t) > y$ and $L_{\max}(\omega^2, t) \leq y$ then $\theta := (\theta, \omega^2)$ and goto Step 1);

Step 3) if $L_{\max}(\omega^1, t) > y$ and $L_{\max}(\omega^2, t) > y$ then $\theta := \oslash$ and RETURN.

Algorithm 3. $\alpha \in [1, \infty), \beta \in [0, 1]$ $O(n^3 \log n)$
$1 \mid d_i \le d_j, d_i - \alpha r_i - \beta p_i \ge d_j - \alpha r_j - \beta p_j \mid L_{\max}$

Step 0) $y := +\infty$; $\pi^* := \omega(N, t)$; $\Phi := \oslash$; $m := 0$; $N' := N \setminus \{\pi^*\}$; $t' := C_{\max}(\pi^*)$; if $N' = \oslash$ then $\Phi := \Phi \cup (\pi^*)$; $m := 1$ and RETURN.

Step 1) if $L_{\max}(\omega^1, t') \le L_{\max}(\pi^*)$ then $\pi^* := (\pi^*, \omega^1)$; $N' := N \setminus \{\pi^*\}$; $t' := C_{\max}(\pi^*)$; goto Step 1);

Step 2) if $(L_{\max}(\omega^1, t') > L_{\max}(\pi^*))\&(L_{\max}(\omega^1, t') < y)$ then $\theta := \theta(N', t', y')$, $y' := L_{\max}(\omega^1, t')$;
if $\theta = \oslash$ then $\pi^* := (\pi^*, \omega^1)$; goto Step 1) else $\pi' := (\pi^*, \theta)$;
if $C_{\max}(\pi_m) < C_{\max}(\pi')$ then $m := m + 1$; $\pi_m := \pi'$; $\Phi := \Phi \cup (\pi_m)$; $y = L_{\max}(\pi_m)$ else $\pi_m = \pi'$; goto Step 1);

Step 3) if $(L_{\max}(\omega^1, t') \ge y)\&$ $(L_{\max}(\omega^2, t') < y)$ then $\pi^* = (\pi^*, \omega^2)$; goto Step 1) else $\pi^* = \pi'_m$ and RETURN.

Pareto optimal schedules for $1 \mid d_i \le d_j, d_i - \alpha r_i - \beta p_i \ge d_j - \alpha r_j - \beta p_j \mid L_{\max}, C_{\max}$

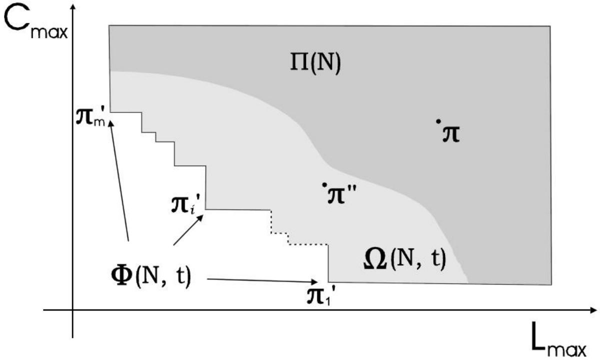
$$\begin{cases} d_1 \le d_2 \le \dots \le d_n; \\ d_1 - \alpha r_1 - \beta p_1 \ge d_2 - \alpha r_2 - \beta p_2 \ge \dots \ge d_n - \alpha r_n - \beta p_n; \\ \alpha \in [1, \infty), \beta \in [0, 1]. \end{cases} \tag{4}$$

$1 \mid d_i \le d_j, d_i - \alpha r_i - \beta p_i \ge d_j - \alpha r_j - \beta p_j \mid L_{\max}, C_{\max}$

$$1 \le \| \Phi(N, t) \| \le n$$

$O(n^3 \log n)$

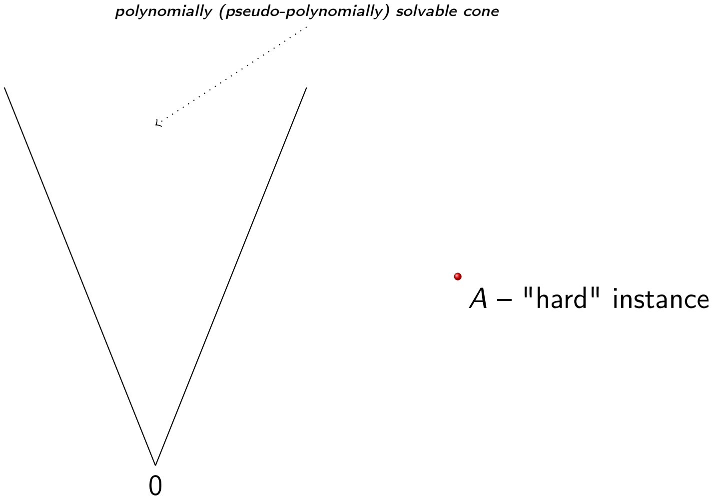
Pareto optimal schedules for $1 \mid d_i \le d_j, d_i - \alpha r_i - \beta p_i \ge d_j - \alpha r_j - \beta p_j \mid L_{max}, C_{max}$



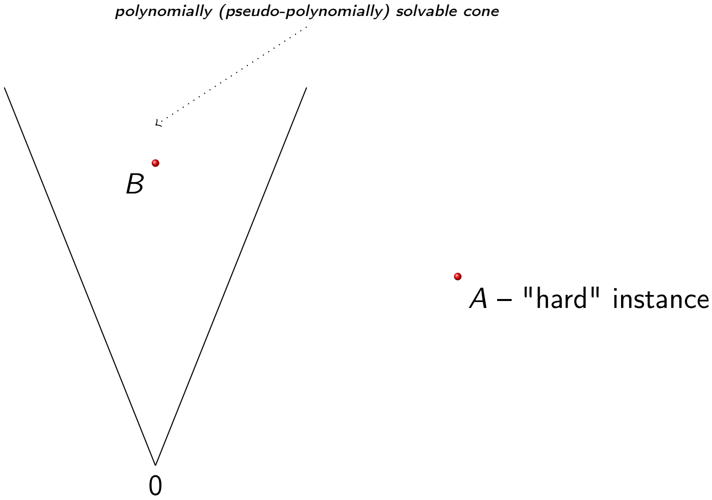
Any instance is point in $m = 3n$-dimension space.

A – "hard" instance

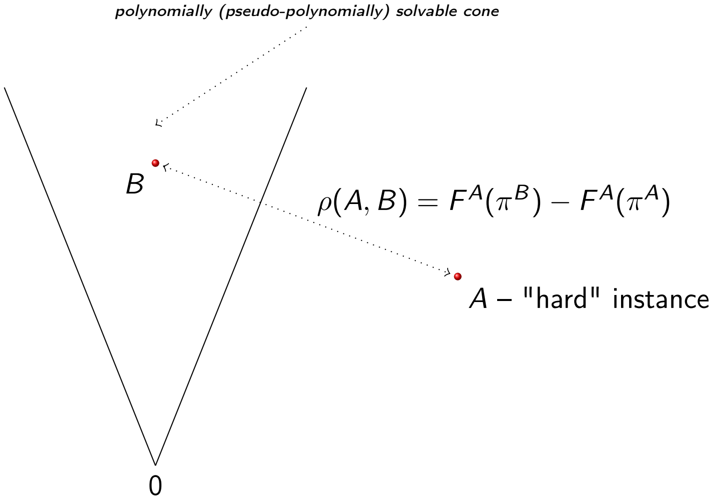
Any instance is point in $m = 3n$-dimension space.



Any instance is point in $m = 3n$-dimension space.



Any instance is point in $m = 3n$-dimension space.



Metric

$1|r_j|L_{max}$

$$\begin{aligned} 0 \le \rho(A,B) = F^A(\pi^B) \quad - \quad & F^A(\pi^A) \le \\ & (\max\{r_j^A - r_j^B\} - \min\{r_j^A - r_j^B\}) + \\ & (\sum |p_j^A - p_j^B|) + \\ & (\max\{d_j^A - d_j^B\} - \min\{d_j^A - d_j^B\}) \end{aligned}$$

Metric

$1|r_j|L_{max}$

$$
\begin{aligned}
0 \le \rho(A,B) = F^A(\pi^B) \quad - \quad & F^A(\pi^A) \le \\
& (\max\{r_j^A - r_j^B\} - \min\{r_j^A - r_j^B\}) + \\
& (\sum |p_j^A - p_j^B|) + \\
& (\max\{d_j^A - d_j^B\} - \min\{d_j^A - d_j^B\})
\end{aligned}
$$

Property of metric

$$\varphi(A) = \max_{j\in N}(r_j^A) - \min_{j\in N}(r_j^A) + \max_{j\in N}(d_j^A) - \min_{j\in N}(d_j^A) + \sum_{j\in N}|p_j^A| \ge 0.$$

$$\begin{cases} \varphi(A) = 0 \iff A \equiv 0; \\ \varphi(\alpha A) = \alpha\varphi(A); \\ \varphi(A+B) \le \varphi(A) + \varphi(B) \end{cases} \tag{5}$$

Metric + Application

$||A|| = \varphi(A)$ $\rho(A, B) = ||A - B||$

Metric + Application

$||A|| = \varphi(A)$ $\rho(A, B) = ||A - B||$

Polynomially (pseudo-polynomially) solvable case

$$\mathcal{A}R + \mathcal{B}P + \mathcal{C}D \leq \mathcal{H}$$

$\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ – matrixes, $R, P, D, \mathcal{H}$ – vectors.

Absolute error approximate solution of the problem $1|r_j|L_{max}$

Polynomially (pseudo-polynomially) solvable case

$$\mathcal{A}R + \mathcal{B}P + \mathcal{C}D \leq \mathcal{H}$$

$\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ – matrixes, $R, P, D, \mathcal{H}$ – vectors.

Absolute error approximate solution of the problem $1|r_j|L_{max}$

Polynomially (pseudo-polynomially) solvable case

$$\mathcal{A}R + \mathcal{B}P + \mathcal{C}D \leq \mathcal{H}$$

$\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ – matrixes, $R, P, D, \mathcal{H}$ – vectors.

Projection of an instance A to a polynomially (pseudo-polynomially) solvable case

The minimum absolute error among all instances from solvable area,– instance B.

$O(n \log n)$

$$\begin{cases} \rho(A,B) = (x_r - y_r) + \sum(x_p - y_p) + (x_d - y_d) \to min \\ y_r \leq r_j^A - r_j^B \leq x_r, \forall\ j; \\ -x_p^j \leq p_j^A - p_j^B \leq x_p^j, \forall\ j, x_p^j \geq 0; \\ y_d \leq d_j^A - d_j^B \leq x_d, \forall\ j; \\ \mathcal{A}R^B + \mathcal{B}P^B + \mathcal{C}D^B \leq \mathcal{H} \end{cases}$$

Linear programming problem

$$\begin{cases} \rho(A,B) = (x_r - y_r) + \sum_j (x_p^j - y_p^j) + (x_d - y_d) \to \min\limits_{\substack{x_r, y_r, x_p^j, x_d, y_d, \\ r_j^B, p_j^B, d_j^B, \forall\, j}} \\ y_r \le r_j^A - r_j^B \le x_r, \forall\, j; \\ -x_p^j \le p_j^A - p_j^B \le x_p^j, \forall\, j, x_p^j \ge 0; \\ y_d \le d_j^A - d_j^B \le x_d, \forall\, j; \\ d_1^B \le d_2^B \le \cdots \le d_n^B; \\ d_1^B - \alpha r_1^B - \beta p_1^B \ge d_2^B - \alpha r_2^B - \beta p_2^B \ge \cdots \ge d_n^B - \alpha r_n^B - \beta p_n^B; \\ \alpha \in [1, \infty), \beta \in [0,1]. \end{cases}$$

$4 + 4n$ variables, $8n - 2$ inequalities $\qquad O(n \log n)$

Any penalties

Any penalties

Initial problem

$$\mu^* = \min_{\pi \in \Pi(N)} \max_{k=\overline{1,n}} \varphi_{j_k}(C_{j_k}(\pi)), \tag{6}$$

Not decreasing functions $\varphi_j(C_j(\pi))$

Any penalties

Initial problem

$$\mu^* = \min_{\pi \in \Pi(N)} \max_{k=\overline{1,n}} \varphi_{j_k}(C_{j_k}(\pi)), \qquad (6)$$

Not decreasing functions $\varphi_j(C_j(\pi))$

Dual problem

$$\nu^* = \max_{k=\overline{1,n}} \min_{\pi \in \Pi(N)} \varphi_{j_k}(C_{j_k}(\pi)). \qquad (7)$$

$r_j = 0, \ \forall \quad j \in N$

Conway R.W., Maxwell W.L., Miller L.W. Theory of Scheduling // Addison-Wesley, Reading, MA. 1967.

Dual problem

$$\nu^* = \max_{k=\overline{1,n}} \min_{\pi \in \Pi(N)} \varphi_{j_k}(C_{j_k}(\pi))$$

Dual problem

$$\nu^* = \max_{k=\overline{1,n}} \min_{\pi \in \Pi(N)} \varphi_{j_k}(C_{j_k}(\pi))$$

$$\nu_k = \min_{\pi \in \Pi(N)} \varphi_{j_k}(C_{j_k}(\pi)), k = 1, 2, \ldots, n. \quad (8)$$

Dual problem

$$\nu^* = \max_{k=\overline{1,n}} \min_{\pi \in \Pi(N)} \varphi_{j_k}(C_{j_k}(\pi))$$

$$\nu_k = \min_{\pi \in \Pi(N)} \varphi_{j_k}(C_{j_k}(\pi)), k = 1, 2, \ldots, n. \tag{8}$$

$$\nu^* = \max_{k=\overline{1,n}} \nu_k. \tag{9}$$

Lemma

$\varphi_j(t), j = 1, 2, \ldots, n$, *any not decreasing functions* $1 \mid r_j \mid \varphi_{\max}$,
$\forall \quad k = 1, 2, \ldots, n, \qquad \nu_n \geq \nu_k, \qquad \nu^* = \nu_n.$

Lemma

$\varphi_j(t), j = 1, 2, \ldots, n,$ *any not decreasing functions* $1 \mid r_j \mid \varphi_{\max}$,
$\forall \quad k = 1, 2, \ldots, n, \qquad \nu_n \geq \nu_k, \qquad \nu^* = \nu_n.$

Algorithm

$\pi^r = (i_1, i_2, \ldots, i_n), \qquad r_{i_1} \leq r_{i_2} \leq \cdots \leq r_{i_n};$
$\pi_k = (\pi^r \setminus i_k, i_k), k = 1, 2, \ldots, n, \qquad \varphi_{i_k}(C_{i_k}(\pi_k));$
$\nu^* = \max\limits_{k=\overline{1,n}} \varphi_{i_k}(C_{i_k}(\pi_k)).$

Lemma

$\varphi_j(t), j = 1, 2, \ldots, n$, *any not decreasing functions* $1 \mid r_j \mid \varphi_{\max}$,
$\forall \quad k = 1, 2, \ldots, n, \qquad \nu_n \geq \nu_k, \qquad \nu^* = \nu_n.$

Algorithm

$\pi^r = (i_1, i_2, \ldots, i_n), \qquad r_{i_1} \leq r_{i_2} \leq \cdots \leq r_{i_n};$
$\pi_k = (\pi^r \setminus i_k, i_k), k = 1, 2, \ldots, n, \qquad \varphi_{i_k}(C_{i_k}(\pi_k));$
$\nu^* = \max\limits_{k=\overline{1,n}} \varphi_{i_k}(C_{i_k}(\pi_k)).$

$O(n^2)$

Initial problem

$$\mu^* = \min_{\pi \in \Pi(N)} \max_{k=\overline{1,n}} \varphi_{j_k}(C_{j_k}(\pi)), \qquad (10)$$

Not decreasing function $\varphi_j(C_j(\pi))$

Initial problem

$$\mu^* = \min_{\pi \in \Pi(N)} \max_{k=\overline{1,n}} \varphi_{j_k}(C_{j_k}(\pi)), \tag{10}$$

Not decreasing function $\varphi_j(C_j(\pi))$

Dual problem

$$\nu^* = \max_{k=\overline{1,n}} \min_{\pi \in \Pi(N)} \varphi_{j_k}(C_{j_k}(\pi)). \tag{11}$$

Initial problem

$$\mu^* = \min_{\pi\in\Pi(N)} \max_{k=\overline{1,n}} \varphi_{j_k}(C_{j_k}(\pi)), \tag{10}$$

Not decreasing function $\varphi_j(C_j(\pi))$

Dual problem

$$\nu^* = \max_{k=\overline{1,n}} \min_{\pi\in\Pi(N)} \varphi_{j_k}(C_{j_k}(\pi)). \tag{11}$$

Theorem

$\varphi_j(t), j = 1, 2, \ldots, n$, *any not decreasing functions* $1 \mid r_j \mid \varphi_{\max}$,
$\forall \ \ k = 1, 2, \ldots, n,$ $\mu^* \geq \nu^*.$

Initial problem

$$\mu^* = \min_{\pi \in \Pi(N)} \max_{k=\overline{1,n}} \varphi_{j_k}(C_{j_k}(\pi)), \quad (10)$$

Not decreasing function $\varphi_j(C_j(\pi))$

Dual problem

$$\nu^* = \max_{k=\overline{1,n}} \min_{\pi \in \Pi(N)} \varphi_{j_k}(C_{j_k}(\pi)). \quad (11)$$

Theorem

$\varphi_j(t), j = 1, 2, \ldots, n$, *any not decreasing functions* $1 \mid r_j \mid \varphi_{\max}$,
$\forall \quad k = 1, 2, \ldots, n,$ $\quad \mu^* \geq \nu^*.$

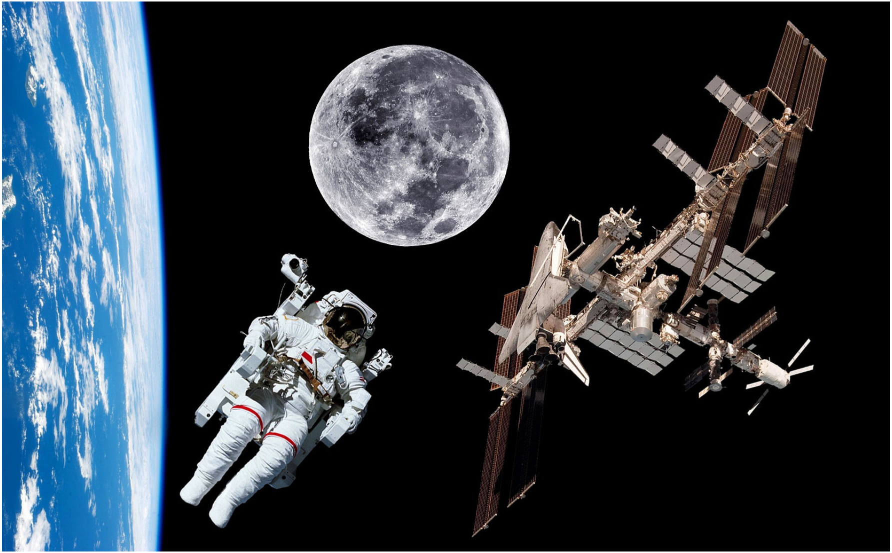
Branch and bounds

Preceding, Dual problem

G	single machine	$O(n^2)$
G	many machines	NP-hard

Not decreasing penalty functions $\varphi_j(C_j(\pi))$

Cosmonauts Training Scheduling Problem



Problem statement

- Set of on-board systems.
- Sets of cosmonauts and crews.
- Set of resources (equipment, teachers, etc.).
- Dates of starts.

It is necessary to prepare appropriate crews to dates of their starts.

Our goals

- to develop mathematical model
- to find approaches to solve it
- to implement Planner system
- to reduce labor costs
- to form new and reschedule available timetable

Cosmonauts Training Scheduling Problem

Mathematical formulation — RCPSP (Resource-Constrained Project Scheduling Problem).

- Resource constraints.
- Precedence constraints.
- More than 4000 publications are devoted to this problem at scholar.google.ru.
- NP-hard in strong sense, there are no pseudo-polynomial algorithms.

Methods for solving RCPSP

- Dynamic programming.
- Methods of Integer Linear Programming.
- Methods of Constraint Programming.
- Heuristic algorithms.

Volume planning problem

Problem statement

- set of on-board systems (near 140);
- required number of cosmonauts of different skills for each on-board system.

Goal: to distribute training qualifications between cosmonauts, minimizing the difference between the maximum and minimum total time of training of cosmonauts.

Results

- heuristic greedy algorithm;
- branch and bound method (CPLEX).

Initial data for volume planning problem

	требемое количество квалификаций							часы на подготовку					
	Корабль К1			*Корабль К2*				*неопытный*			*опытный*		
	С	О	П	С	О	П		С	О	П	С	О	П
Срочное покидание в аварийных ситуациях	0	3	0	0	3	0		23	23	22	23	23	22
Система инвентарного учета	0	0	3	0	1	2		0	17	2	0	9	0
Информационно-управляющая система	2	0	0	1	0	0		12	12	1	4	4	1
Бортовая вычислительная система	1	0	2	1	0	2		15	11	5,5	2	2	2
Система управления бортовым комплексом/ бортовой аппаратурой	1	0	2	1	0	2		28	22	9,5	2	2	2
Система бортовых измерений	1	0	2	1	0	2		15	13	2	4	4	0
Средства радиосвязи	1	1	1	1	0	2		35	28	11,25	4	4	2
Телевизионная система	1	0	2	1	0	2		11	11	2	4	4	0
Система обеспечения жизнедеятельности	1	1	1	1	0	2		70	57,25	26	12	12	5
Система энергоснабжения	1	0	2	1	0	2		20	18	2	8	6	0
Система управления движением и навигацией	1	1	1	1	1	1		38	17,5	2	8	8	1
Двигательные установки	1	0	0	1	0	0		4	0	0	2	0	0
Оптико-визуальные системы	1	0	0	1	0	0		0	4	0	0	2	0
Курс	1	0	0	1	0	0		7	7	0	2	2	0
Система стыковки	1	0	2	1	0	2		16	13	4	4	4	2
Конструкция и компоновка	0	2	1	0	2	1							
Система обеспечения теплового режима	1	1	1	1	0	2		43	24	6,5	8	8	2
Фотоаппаратура	0	1	2	0	1	2		0	19	0	0	8	0
Видеоаппаратура, аудиоаппаратура	0	1	2	0	1	2		0	28	0	0	12	0
ЛРС	1	0	0	1	0	0		22	14	5	11	8	5
Оборудование для ВКД (скафандр Орлан, шлюзовой отсек, инструменты для ВКД)	0	2	0	0	1	0		0	60	8	0	32	8

The experimental results for volume planning problem

№	Опыт	Жадный алгоритм			CPLEX		
		max	min	δ	max	min	δ
1	3 неоп.	889.5	887.0	2.5	888.05	887.75	0.3
	3 оп.	570.5	569	1.5	570	569.5	0.5
	1 оп., 2 неоп.	721.7	694.5	27.2	697.25	695.25	2
	2 оп., 1 неоп.	669.7	598.0	71.7	616.5	612.75	3.75
2	3 неоп.	266.25	265	1.25	265.75	265.2	0.55
	3 оп.	234.2	233	1.2	233.75	233.25	0.5
	1 оп., 2 неоп.	245.5	244.0	1.5	244.45	244	0.45
	2 оп., 1 неоп.	235.0	233.25	1.75	233.75	233.25	0.5
3	3 неоп.	660.2	659.5	0.7	659.85	659.75	0.1
	3 оп.	353.5	353.05	0.45	353.5	353	0.5
	1 оп.,2 неоп.	497.95	493.5	4.45	484.05	481.75	2.3
	2 оп., 1 неоп.	398.05	394.0	4.05	393.5	392.5	1
4	3 неоп.	925.75	924.2	1.55	925	924.8	0.2
	3 оп.	587	586.5	0.5	587	586.5	0.5
	1 оп., 2 неоп.	774.5	694.5	80.0	731.5	730.75	0.75
	2 неоп., 1 оп.	649.2	648.5	0.7	628.75	628	0.75

Measure of unsolvability

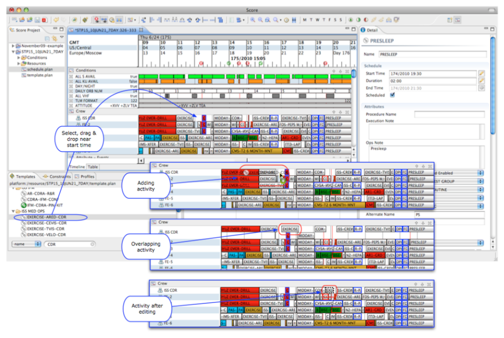
Timetabling problem

- Planing horizon is about 3 years.
- Each cosmonaut has an individual learning plan.
- 10 crews are studying simultaneously.
- There are main and backup crews.

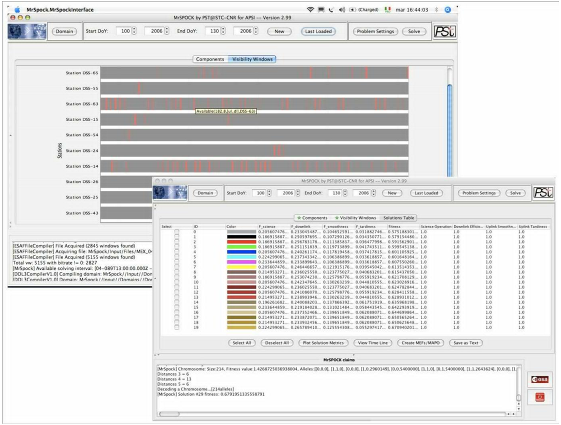
BQ	BT	BU	BV	BW	BX	CA	CB	CC	CD	CE	CH	CI	CJ
W:10 D:5	W:11 D:1	W:11 D:2	W:11 D:3	W:11 D:4	W:11 D:5	W:12 D:1	W:12 D:2	W:12 D:3	W:12 D:4	W:12 D:5	W:13 D:1	W:13 D:2	W:13 D:3
ТПК СУС С1 БК#7	ТПК СУС Л ОРРЕЗСП БК#7	ТПК СУС С2 БК#7	ТПК СУС Л НАДБЕЗ2 БК#7	ТПК СУС С3 БК#7	ТПК СУС Л РАБАВНОС БК#7	ТПК СУС ПЗ ЦШТСП БК#7	ТПК СУС ПЗ РЕЗСП БК#7	ТПК СУС К1 БК#7	ТПК СУС К2 БК#7	ТПК СУД СБЛ Л ОРГРЕЖ БК#8	ТПК СУД СБЛ Л КОРРЦМ БК#8	ТПК СУД СБЛ Л РАБКУРС БК#8	ТПК СУД СБЛ Л ДФРС БК#8
	ТПК ССВП Л ПЛУСИС БК#10	ТПК СУС Л НАДБЕЗ1 БК#7			ТПК ССВП Л РЕЗРЕЖРАБ БК#10				ТПК СУС Э БК#7	Физ-ра		ТПК СУД СБЛ Л ОРЛСК БК#8	Англ.яз
Обед	Обед	Обед	Обед	Обед	Обед	Обед	Обед	Обед	Обед	Обед	Обед	Обед	Обед
	ТПК СУС Л ОРСРСП БК#7	ТПК ССВП Л КОМУПР БК#10		ТПК ССВП Л ШТРЕЖРАБ БК#10	ТПК ССВП Л НШС БК#10	ТПК ССВП ПЗ РАБССВП БК#10	Англ.яз	ТПК ССВП С БК#10		ТПК ССВП К БК#10	Англ.яз	ТПК СКГС Л ОБЩСВ БК#10	ТПК СКГС ПЗ РБКЛ БК#10
Физ-ра	Англ.яз	Англ.яз	Физ-ра	Физ-ра		Англ.яз	ТПК ССВП ПЗ РУЧОП БК#10	Физ-ра	ТПК СУД СБЛ Л ОС БК#8		ТПК ССВП Э БК#10	Физ-ра	ТПК СКГС К БК#10 ТПК СМО ПЗ БК#20
ТПК СУС С1 БК#7	ТПК СУС Л ОРРЕЗСП БК#7	ТПК СУС С2 БК#7	ТПК СУС Л НАДБЕЗ2 БК#7	ТПК СУС С3 БК#7	ТПК СУС Л РАБАВНОС БК#7	ТПК СУС ПЗ ЦШТСП БК#7	ТПК СУС ПЗ РЕЗСП БК#7	ТПК СУС К1 БК#7	ТПК СУС К2 БК#7	ТПК СУД СБЛ Л ОРГРЕЖ БК#8	ТПК СУД СБЛ Л КОРРЦМ БК#8	ТПК СУД СБЛ Л РАБКУРС БК#8	ТПК СУД СБЛ Л ДФРС БК#8
	ТПК ССВП Л ПЛУСИС БК#10	ТПК СУС Л НАДБЕЗ1 БК#7			ТПК ССВП Л РЕЗРЕЖРАБ БК#10				Физ-ра			ТПК СУД СБЛ Л ОРЛСК БК#8	Физ-ра
Обед	Обед	Обед	Обед	Обед	Обед	Обед	Обед	Обед	Обед	Обед	Обед	Обед	Обед
	ТПК СУС Л ОРСРСП БК#7	ТПК ССВП Л КОМУПР БК#10	Англ.яз	ТПК ССВП Л ШТРЕЖРАБ БК#10	ТПК ССВП Л НШС БК#10	ТПК ССВП ПЗ РАБССВП БК#10	Англ.яз	ТПК ССВП С БК#10	ТПК СУС Э БК#7	ТПК ССВП К БК#10	ТПК ССВП Э БК#10 Англ.яз	ТПК СКГС Л ОБЩСВ БК#10	ТПК СКГС ПЗ РБКЛ БК#10
Физ-ра	Англ.яз	Физ-ра		Физ-ра		Англ.яз	ТПК ССВП ПЗ РУЧОП БК#10	Физ-ра	ТПК СУД СБЛ Л ОС БК#8			Физ-ра	ТПК СКГС К БК#10 ТПК СМО ПЗ БК#20
	Англ.яз	ТПК КИК ПЗ ОПЕР БИ2 БК#1	ТПК КИК К БИ2 БК#1	ТПК КИК ПЗ ПРИМ БК#1	ТПК СРС Л УПРАВЛЕН БК#13	ТПК СУД АК Л ОСАК БК#5	ТПК СУД АК ПЗ НБИ2 БК#5	ТПК СУД ДК ППЗ НБИ2 БК#6	ТПК СУД ДК ПЗ НБИ2 БК#6	ТПК СУС Л НБИ2 БК#7	ТПК СУС Л ЦШС БК#7	ТПК СУС ПЗ НБИ2 БК#7	ТПК КСОЖ Л СОГС БК#12
	ТПК КИК Л ОЗН БИ2 БК#1	ТПК ПК Л НАЗН БК#2	Физ-ра		ТПК СРС Л ОРГАНРСВ БК#13	ТПК СУД АК Л РРАК БК#5	Физ-ра	ТПК СРС ПЗ БК#13		ТПК СУС Л АСАБНО БК#7			
Обед	Обед	Обед	Обед	Обед	Обед	Обед	Обед	Обед	Обед	Обед	Обед	Обед	Обед
	ТПК КИК Л ОТС БИ2 БК#1	Англ.яз	ТПК ПК З БК#2 ТПК КИК З БК#1	ТПК СРС Л НАЗНАЧ БК#13	ТПК КДУ Л НБИ2 БК#3	ТПК СРС Л НШС БК#13	ТПК СУД ДК Л НБИ2 БК#6	Англ.яз	ТПК СРС Э БК#13 ТПК СРС ПЗ ИРИДИУМ БК#13		Англ.яз	ТПК СУД СБЛ Л ОП БИ2 БК#8	ТПК СУД СБЛ Л ДУ БИ2 БК#8
	ТПК ПК Л ОБЩПРЕД БК#2	ТПК ПК ПЗ РАБПУЛ БК#2 ТПК ПК К БК#2		Физ-ра		Англ.яз		ТПК СРС К ПРЕДСТРАЗ БК#13	Физ-ра		ТПК КСОЖ Л ВВЕДЕНИЕ БК#12	Физ-ра	

Review of other space agencies systems

NASA – TAMS, FOCAS, STAR



KAREN AU, SAMUEL SANTIAGO, RICHARD PAPASIN, MAY WINDERM, TRISTAN LE. *Streamlining Space Training Mission Operations with Web Technologies. An Approach to Developing Integral Business Applications for Large Organizations* // IEEE 4th International Conference on. Space Mission Challenges for Information Technology (SMC-IT), 2011, pp.159-166.



SPAGNULO, M., FLEETER, R., BALDUCCINI, M., NASINI, F. *Space Program Management : Methods and Tools* // Spagnulo, M., Fleeter, R., Balduccini, M., Nasini, F., Springer-Verlag New York - 2013. - 352 c.

Problem statement

K – a number of cosmonauts;

J_k – each cosmonaut k has his own set of training tasks;

p_j – execution time of task $j \in J$;

R – set of resources.

The goal is

to form a training schedule for each cosmonaut

Time intervals

W – set of planning weeks, where $|W| = 156$ weeks (3 years);

$D_w = \{1,2,3,4,5\}$ – set of work days per week, $w \in W$;

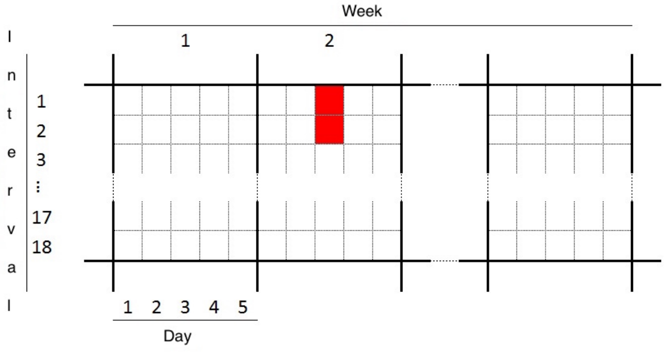
$H_{wd} = \{1, ..., 18\}$ – set of half-hour intervals of day $d \in D_w$ of week $w \in W$.

$$Y = \{(w, d, h) | w \in W, d \in D_w, h \in H_{wd}\}, \quad |Y| \approx 14040$$

$t(w, d, h)$ – considering time moment.

Variables

$$x_{jwdh} = \begin{cases} 1, & \text{iff task } j \text{ is started} \\ & \text{from interval } h \text{ of day } d \text{ of week } w; \\ 0, & \text{else.} \end{cases}$$



Constraints

Precedence relations between the tasks (academic plan)

$$\sum_{(w,d,h)\in Y} t(w,d,h)(x_{j_2 wdh} - x_{j_1 wdh}) \geq p_{j_1}, \tag{12}$$

$$\forall (j_1, j_2) \in \Gamma_k.$$

The resource limits (teachers, simulators, trainers)

$$\sum_{j\in J} rc_{jr} \sum_{\substack{h' > 0, \\ h - p_j + 1 \leq h' \leq h}} x_{jwdh'} \leq ra_{rwdh}, \tag{13}$$

$$\forall r \in R,\ \forall (w,d,h) \in Y. \qquad |Y| \approx 14040,\ |R| \approx 100.$$

Constraints

No more than ... (frequency of classes)

$$\sum_{j \in J^F} \sum_{d \in D_w} \sum_{h \in H_{wd}} x_{jwdh} \leq 2, \quad \forall w \in W. \tag{14}$$

Each cosmonaut may have no more than 2 physical trainings per week.

Excluding some time intervals

$$\sum_{j \in J_{[h_1;h_2]}} \sum_{h_1 - p_j + 1 \leq h \leq h_2} x_{jwdh} = 0, \tag{15}$$

$$\forall w \in W, \quad \forall d \in D_w;$$

$[h_1; h_2]$ – time period when performing task j is forbidden.

It is forbidden to practice in the hyperbaric chamber after lunch.

Comparison of two approaches to solving the scheduling problem for 1 crew

N	CPLEX MIP				CPLEX CP			
	Time, c	Var.	Constr.	Iter.	Time, c	Var.	Constr.	Branch.
1	09.06	26820	37620	21922	0.250	291	2170	1272
2	30.75	52680	60066	54234	0.329	363	2788	1512
3	559.84	73500	87846	5019412	0.438	492	3548	2008
4	375.834	108720	121578	2032790	0.703	606	4263	2784
5	374.63	115200	125466	2022320	0.610	642	4348	2912
7	346.30	144480	157920	820534	0.640	654	4374	2648
10	6657.98	204000	210646	16 917 014	1.317	852	5738	3 448

N is a number of on-board systems.

Conclusion

Our results

Schedule for 1 crew for 1 year 3 moths

Our plans

Schedule for 2 crew for 2 year

Railway scheduling pioneers

Frank, O., Two-Way Traffic on a Single Line of Railway, Oper. Res., 1966, vol. 14, no. 5, pp. 801–811.

Szpigel, B., Optimal Train Scheduling on a Single Line Railway, Oper. Res., 1973, pp. 344–351.

Relation between railway planning problems and classical scheduling problems

- track segments = «machines»
- trains = «jobs»

Existing approaches and solution methods

1. Considering in terms of job-shop.

Szpigel B. Optimal train scheduling on a single line railway. Oper Res, 344 - 351, 1973.

Sotskov Y. Shifting bottleneck algorithm for train scheduling in a single-track railway. Proccedings of the 14th IFAC Symposium on Information Control Problems. Part 1. Bucharest/Romania. 87 - 92. 2012.

Gafarov E.R., Dolgui A., Lazarev A.A. Two-Station Single-Track Railway Scheduling Problem With Trains of Equal Speed. Computers and Industrial Engineering. 85:260 - 267. 2015.

Harbering J., Ranade A., Schmidt M. Single Track Train Scheduling. Institute of Numerical and Applied Mathematics. preprint. 18. 2015.

Existing approaches and solution methods

2. Integer linear programming

Brannlund U., Lindberg P.O, Nou A. and Nilsson J.E.
Railway Timetabling Using Lagrangian Relaxation.
Transportation Science 32(4):358 - 369. 1998.

Lazarev, A.A. and Musatova, E.G.
Integer Formulations of the Problem of Railway Train Formation and Timetabling,
Upravlen. Bol'shimi Sist., 2012, no. 38, pp. 161–169.

3. Heuristics

Sotskov Y.
Shifting bottleneck algorithm for train scheduling in a single-track railway.
Proccedings of the 14th IFAC Symposium on Information Control Problems. Part 1. Bucharest/Romania. 87 - 92. 2012.

Mu S., Maged D.
Scheduling freight trains traveling on complex networks.
Transportation Research Part B: Methodological. 45(7):1103 - 1123. 2011.

Carey M., and Lockwood D.
A model, algorithms and strategy for train pathing.
The Journal of Operational Research Society. 8(46):988 - 1005. 1995.

Exicting approaches and solution methods

Allocation of polynomially solvable cases of railway scheduling problems

Gafarov E.R., Dolgui A., Lazarev A.A.
Two-Station Single-Track Railway Scheduling Problem With Trains of Equal Speed.
Computers and Industrial Engineering. 85:260 - 267. 2015.

Harbering J., Ranade A., Schmidt M.
Single Track Train Scheduling.
Institute of Numerical and Applied Mathematics. preprint. 18. 2015.

Disser Y., Klimm M., Lubbecke E.
Scheduling Bidirectional Traffic on a Path.
In Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming (ICALP). 406 - 418. 2015.

Laboratory projects in railway scheduling

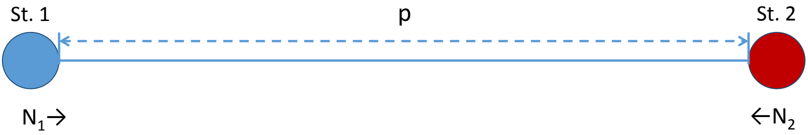
Small-scale problems

- Scheduling problem on single railway tracks.
- Goal – the development of exact polynomially solvable algorithms with small computational complexity.
- Solution approach – dynamical programming.

Large-scale problems

- The freight car routing problem.
- Goal – the construction of operational plan with feasible solution time.
- Solution approach – integer linear programming, LP-relaxation, column generation.

Single track railway scheduling problem



Initial data

- $|N_1| = n$, $|N_2| = n'$, $N = N_1 \cup N_2$, $|N| = n + n'$.
- All trains have equal speed, track traversing time – p.
- Minimal time between the departure of two trains from one station – β.
- The transportation starts at time $t = 0$.

Denote the problem as $STR2$ (Single Track Railway Scheduling Problem).

Problem formulation

Schedule

In schedule σ, for each train $i \in N$

$S_i(\sigma)$ – it's departure time;

$C_i(\sigma)$ – arrival time, $C_i(\sigma) = S_i(\sigma) + p$.

Objective function

Family of objective functions.

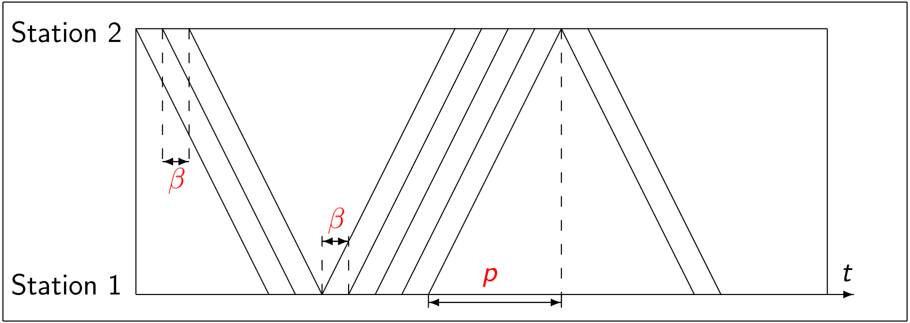
The approach will be demonstrated on the maximum lateness objective function $L_{max}(\sigma)$,

$$L_{max}(\sigma) = \max_{i \in N} L_i = \max_{i \in N} \{C_i(\sigma) - d_i\}.$$

Dynamic programming approach

Assumption

We will consider schedule schedule σ which possess the following property: for any point in time t such that $0 \leq t \leq C_{max}(\sigma)$ there exists at least one train $i \in N$ satisfying the condition $S_i(\sigma) \leq t \leq C_i(\sigma)$.



Dynamic programming approach

Assumption

Train departure order is specified.

Maximum lateness L_{max}

For objective function $L_{max}(\sigma) = \max_{i \in N}\{C_i(\sigma) - d_i\}$ there exists an optimal schedule σ in which trains depart from each station in a nondecreasing order of due dates d_i.

Numbering of trains

On each station trains are numbered in the decreasing order of their departure times, $i > j$ implies that, in any schedule σ, $S_i(\sigma) < S_j(\sigma)$.

Subproblem $\mathbf{P}(\boldsymbol{k_1}, \boldsymbol{k_2}, \boldsymbol{s})$

set of unsent trains
on station 1,
$k_1 \in \{0,1,2,\dots,n\} \in N_1$

additional condition:
first train depart from
station $s \in \{1,2\}$

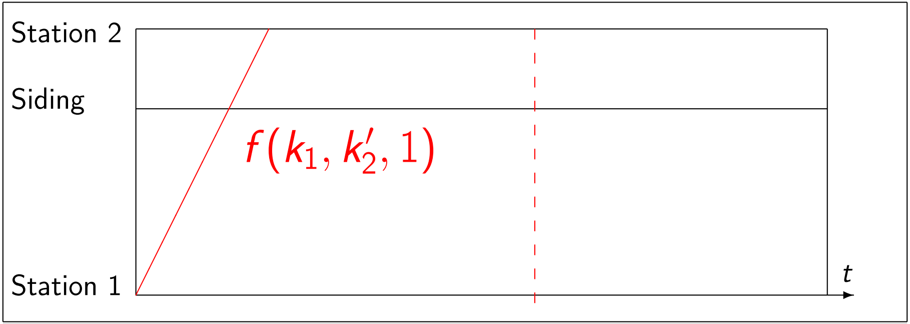
set of unsent trains on station 2,
$k_2 \in \{0,1,2,\dots,n'\} \in N_2$

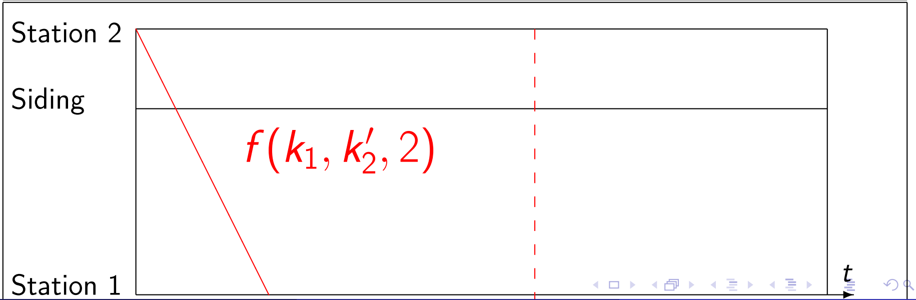
Optimal value of the objective function for $P(k_1, k_2', s)$

$$f(k_1, k_2', s) = F(\sigma^*),$$

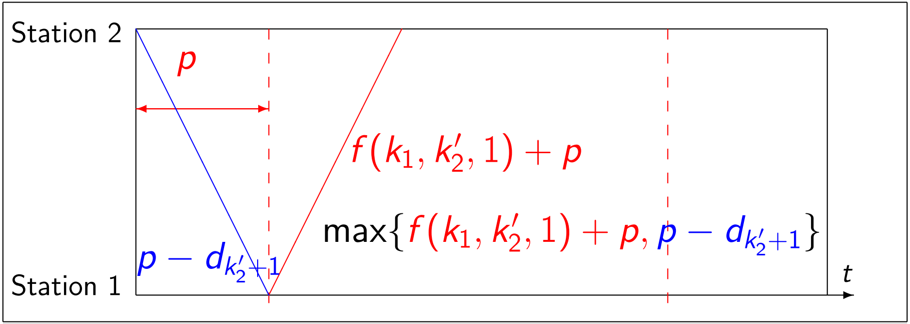
where σ^* is an optimal schedule for $P(k_1, k_2', s)$.

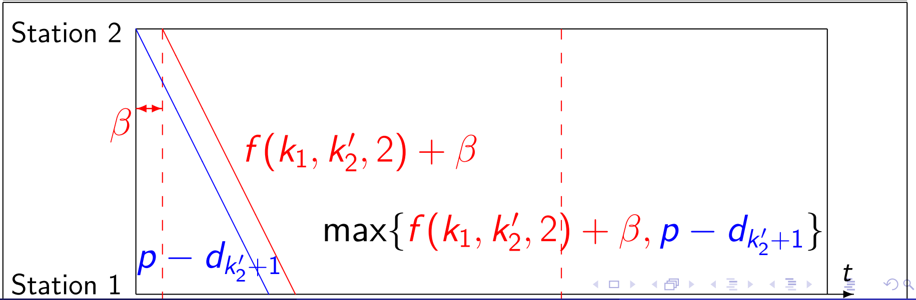
Solution algorithm





Solution algorithm





Dynamic programming approach

$$f(k_1, k_2' + 1, 2) = \max \begin{cases} p - d_{k_2'+1}; \\ \min \begin{cases} f(k_1, k_2', 1) + p; \\ f(k_1, k_2', 2) + \beta; \end{cases} \end{cases}$$

$$\text{for each } k_2' \in \{1', ..., n' - 1'\}, \; k_1 \neq 0.$$

Dynamic programming approach

Setting

$$f(1, 0', 1) = p - d_1$$

$$f(0, 1', 2) = p - d_{1'}$$

Bellman equation

$$f(k_1+1, k_2', 1) = \max \begin{cases} p - d_{k_1+1}; \\ \min \begin{cases} f(k_1, k_2', 1) + \beta; \\ f(k_1, k_2', 2) + p. \end{cases} \end{cases} \qquad k_1 \in \{1, ..., n-1\},\ k_2' \neq 0'$$

$$f(k_1, k_2'+1, 2) = \max \begin{cases} p - d_{k_2'+1}; \\ \min \begin{cases} f(k_1, k_2', 1) + p; \\ f(k_1, k_2', 2) + \beta. \end{cases} \end{cases} \qquad k_2' \in \{1', ..., n'-1'\},\ k_1 \neq 0$$

Dynamic programming approach

Optimal objective function value of the original problem

$$\min\{f(n, n', 1), f(n, n', 2)\}$$

Computational complexity

$$O((n + n')^2)$$

Value of $f(k_1, k_2', s)$ is computed for:

each pair of k_1, $k_1 \in \{1, ..., n\}$), and k_2', $k_2 \in \{1, ..., n'\}$.

Dynamic programming approach

Other objective functions

This solution procedure can applied to a set of objective functions, for example for

$$\sum w_i C_i(\sigma) = \sum_{i \in N} w_i C_i(\sigma)$$

Condition

"Shifted" schedule σ_t of schedule σ, $C_i(\sigma) - C_i(\sigma_t) = t$ for all $i \in N$.

There exists $G(k_1, k_2', s)$ so that $F(\sigma_t) = F(\sigma) + G(k_1, k_2', t)$.

for L_{max}: $G(k_1, k_2', t) = t$;

for $\sum w_i C_i(\sigma)$: $G(k_1, k_2', t) = \sum_{i=1}^{k_1} w_i t + \sum_{j=1'}^{k_2'} w_j t$.

Dynamic programming approach

General form of objective functions

$$\bigodot_{i \in N} \varphi_i(C_i(\sigma)),$$

where

$\varphi_i(\cdot)$ – nondecreasing function, defined for each train $i \in N$,

$\odot$ – some commutative and associative operation such,

for any numbers a_1, a_2, b_1, b_2, $\odot$ satisfy $a_1 \leq a_2$ and $b_1 \leq b_2$,

$$a_1 \odot b_1 \leq a_2 \odot b_2.$$

Dynamic programming approach

Solution procedure

$$STR2||\bigodot_{i\in N}\varphi_i(C_i(\sigma))$$

Specified train departure order on each station.

Polynomial set of possible departure times T, $|T| = O((n+n')^2)$.

Subproblem: $P(k_1, k_2', s, t)$, $f(k_1, k_2', s, t)$ is calculated for

each pair of k_1, $k_1 \in \{1, ..., n\}$;
each pair of k_2', $k_2 \in \{1, ..., n'\}$;
all $t \in T$.

Computational complexity – $O((n+n')^4)$.

Dynamic programming approach

Minimization of maximum cost functions

$$F_{max}(\sigma) = \max_{i \in N} \varphi_i(C_i(\sigma))$$

No specified order of train departure on each station.

Iterative optimization procedure

dynamic programming algorithm for $STR2||L_{max}$

general optimisation scheme, presented by Zinder and Shkurba[1]

[1] Zinder, Y. and Shkurba, V. Effective iterative algorithms in scheduling theory. Cybernetics, 21(1), 86–90. 1985.

Iterative optimisation procedure

Algorithm 1 Solution method for the train scheduling problem $STR2||F_{max}$

1: $V := \max_{i \in N} \varphi_i(p)$ (lower bound)

2: **for** $i := 1$ **to** $n + n'$ **do** — Due date setting

3:    **if** $\varphi_i(\tau_r) \le V$ **then**

4:       $d_i := \tau_r$

5:    **else**

6:       choose τ_k so that $\varphi_i(\tau_k) \le V < \varphi_i(\tau_{k+1})$

7:       $d_i := \tau_k$

8:    **end if**

9: **end for**

10: construct schedule σ by solving $STR2||L_{max}$

11: $L := L_{max}(\sigma)$

12: **if** $L > 0$ **then** — Lower bound checking

13:    $V := \min_{i \in \{j:\, j \in N,\, (d_j + L) \in T'\}} \varphi_i(d_i + L)$ (lower bound)

14:    **go to** 2

15: **else**

16:    **return** σ is an optimal value

17: **end if**

Computational complexity

$O((n + n')^5 \log(n + n'))$

Conclusion

Dynamic programming procedure for a set of objective functions

$$F(\sigma) = \bigodot_{i \in N} \varphi_i(C_i(\sigma))$$

Computational complexity is $O((n+n')^4)$,
can be reduced for a subset of objective functions – $O((n+n')^2)$.

Iterative optimisation procedure for maximum cost functions

$$F_{max}(\sigma) = \max_{i \in N} \varphi_i(C_i(\sigma))$$

Computational complexity is $O((n+n')^5 \log(n+n'))$.

Single track railway scheduling problem

Solution algorithm complexity

Problem	Complexity
$STR2\,\|\,\|L_{max}$	$O(n^2)$
$STR2\,\|\,\|\sum w_j C_j$	$O(n^2)$
$STR2\,\|\,\|\max\limits_{j\in N}\varphi_j(C_j(\sigma))$	$O(n^5\log n)$
$STR2\,\|p(j),\lambda\|L_{max}$	$O(n^\lambda)$
$STR2\,\|p(j),\lambda\|\sum w_j C_j$	$O(n^\lambda)$
$STR2\,\|p(j),\lambda\|\sum U_j(\sigma)$	$O(n^{2\lambda})$
$STR2\,\|p(j),\lambda\|\bigodot\limits_j \varphi(C_j)$	$O(n^{\alpha^2+\alpha}n^\lambda)$
$STR2\,\|p(j),\lambda,V\|\max\limits_{j\in N}\varphi_j(C_j(\sigma))$	$O(q^2\log qn^{2\alpha^2+2\alpha+1}n^\lambda\log n)$

λ – the number of subsets with possible fixed departure order $p(j)$ – different train traversing times V – feasible intervals of movement

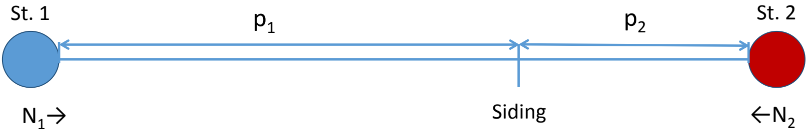
Single track railway scheduling problem with a siding

What is the siding?

Main track

Additional track

Single track railway scheduling problem with a siding



Initial data

- One siding, capacity is one train.
- $|N_1| = n_1$, $|N_2| = n_2$, all trains have equal speed.
- Traversing times: p_1, p_2, $p_1 \geq p_2$.
- For each train i from station s, $i \in N_s$, $s \in \{1, 2\}$, due date d_s^i and cost coefficient w_s^i are given;
- Release times: $r_s^i = 0$, $i \in N_s$, $s \in \{1, 2\}$.

Denote the problem as STRSP2 (Single Track Railway Scheduling Problem).

Single track railway scheduling problem with a siding

Schedule

We need to construct optimal schedule σ, i.e. to set for each train number i moving from station s, $i \in N_s$, $s \in \{1, 2\}$, it's departure time $S_s^i(\sigma)$, stop time in the siding $\tau_s^i(\sigma)$ and arrival time $C_s^i(\sigma)$.

Objective function

Minimizing maximum lateness

$$L_{max} = \max_{i \in N_s, s \in \{1,2\}} \{L_s^i\},$$

where

$$L_s^i = C_s^i - d_s^i,$$

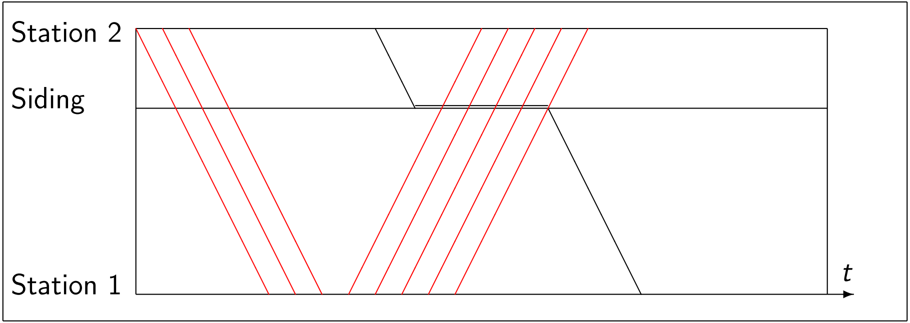
and weighted sum of arrival moments

$$\sum w_j C_j = \sum_{i \in N_s,\, s \in \{1,2\}} w_s^i C_s^i.$$

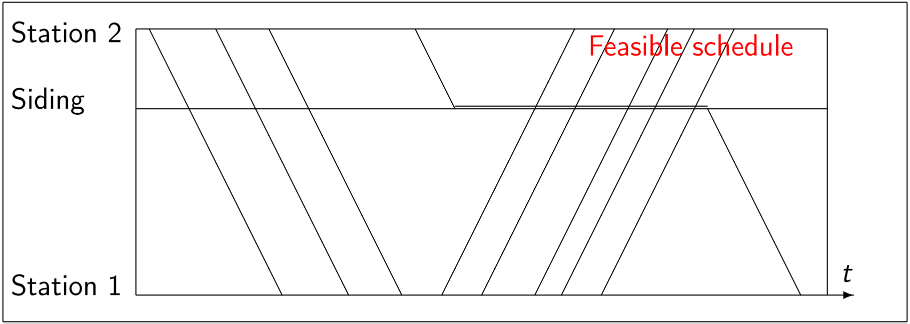
Schedule properties for presented model

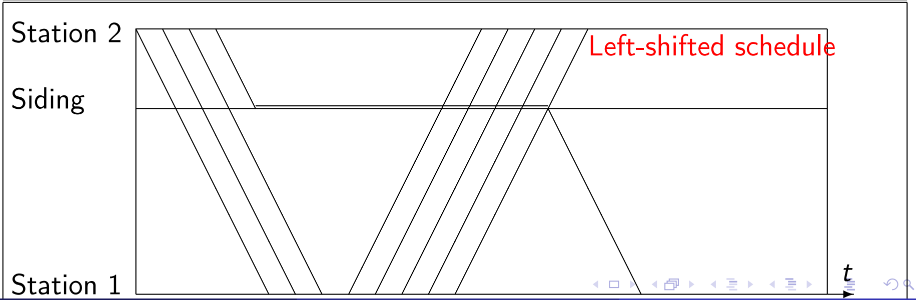
Express

Express is the train i moving from station s, $i \in N_s$, $s \in \{1, 2\}$, if it doesn't stop in the siding, i.e. $\tau_s^i = 0$.

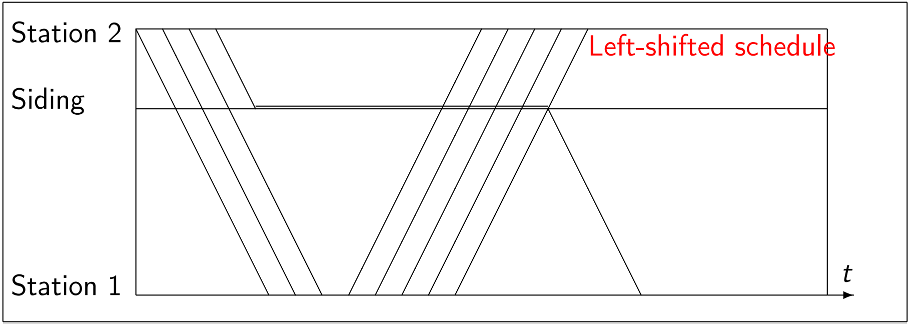


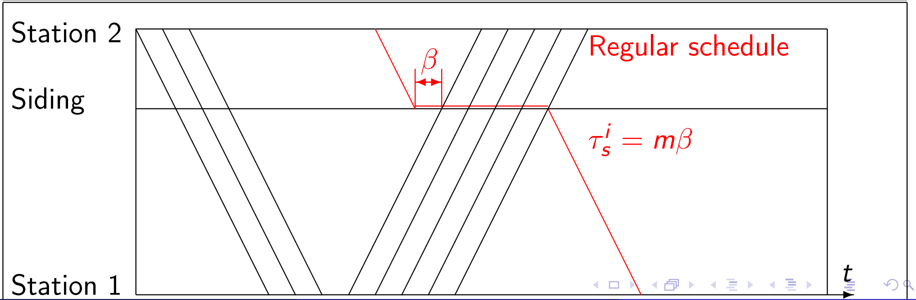
Schedule properties for presented model



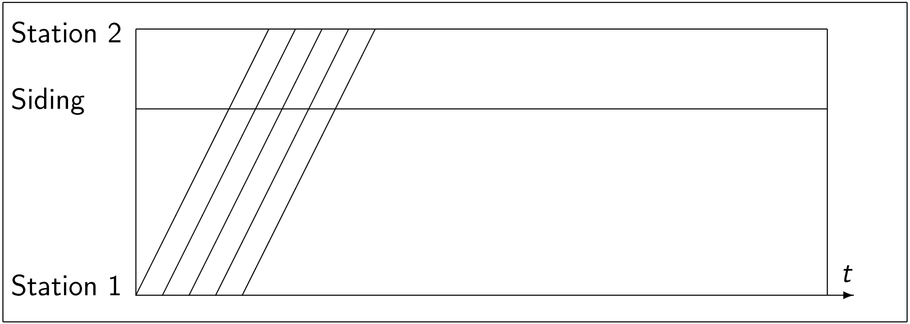


Schedule properties for presented model



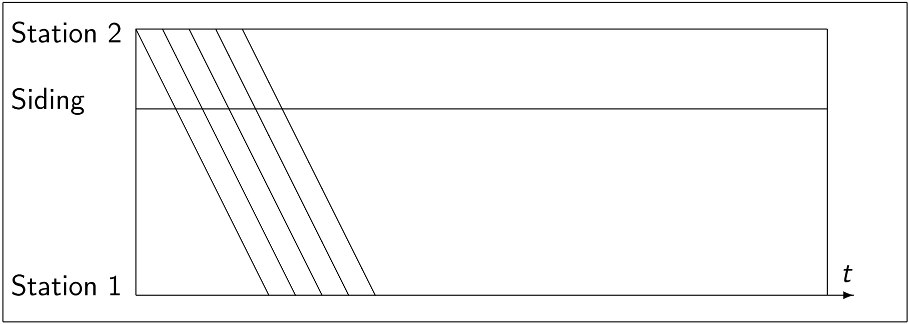


States



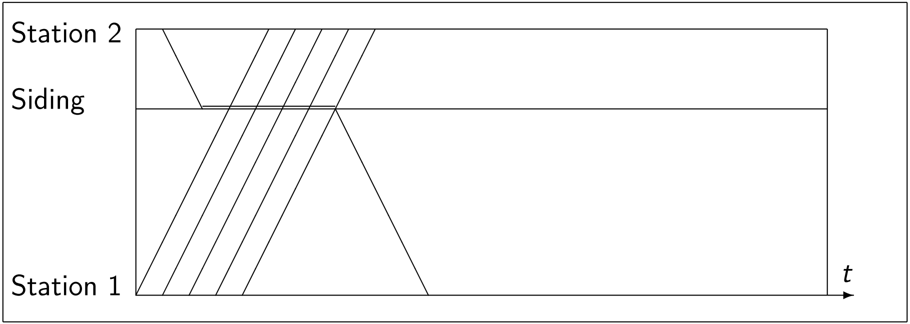
1) Batch moving from station 1 with empty siding.

States



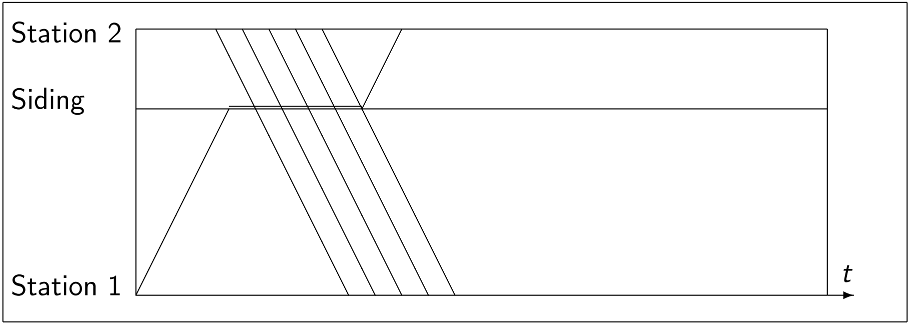
2) Batch moving from station 2 with empty siding.

States



3) Batch moving from station 1 with occupied siding.

States



4) Batch moving from station 2 with occupied siding.

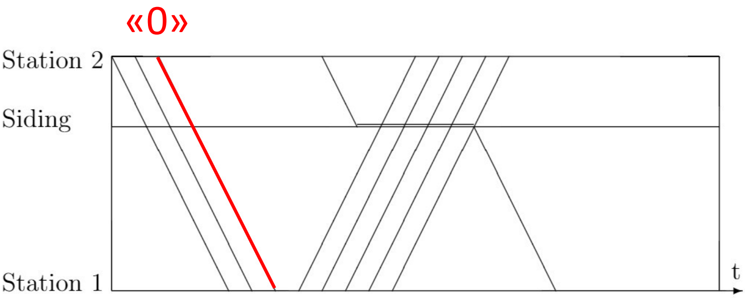
States

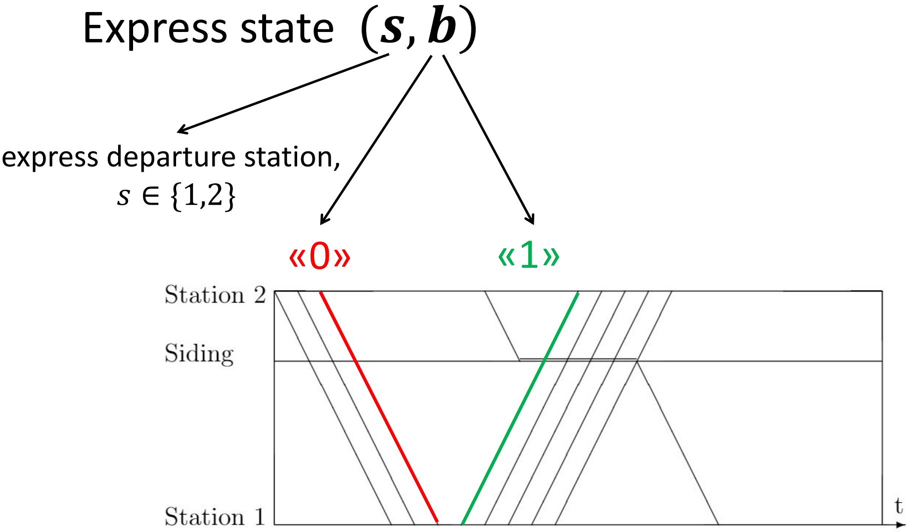
Express state $(\boldsymbol{s}, \boldsymbol{b})$

express departure station,
$s \in \{1,2\}$

Express state $(\boldsymbol{s}, \boldsymbol{b})$

express departure station,
$s \in \{1,2\}$

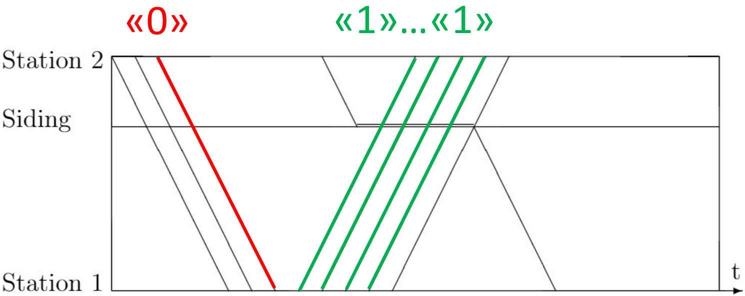



Express state $(\boldsymbol{s}, \boldsymbol{b})$
express departure station,
$s \in \{1,2\}$
«0»
«1»
Station 2
Siding
Station 1
t

States

Express state $(\boldsymbol{s}, \boldsymbol{b})$

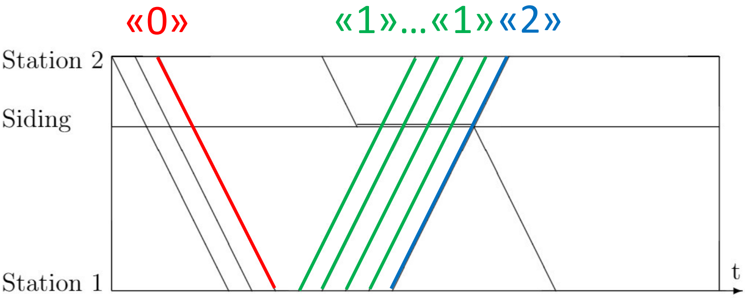
express departure station,
$s \in \{1,2\}$



States

Express state $(\boldsymbol{s}, \boldsymbol{b})$

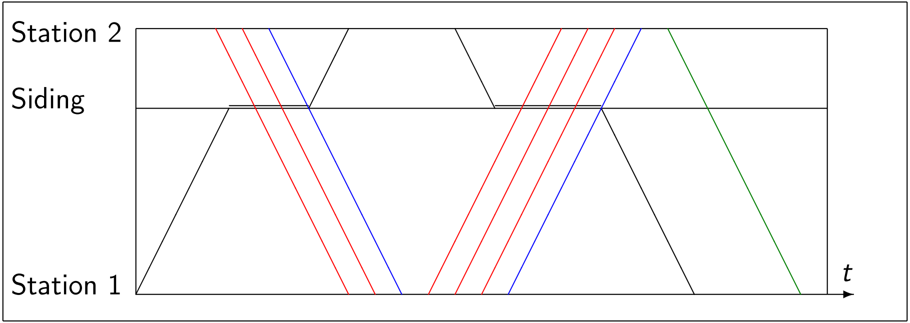
express departure station,
$s \in \{1,2\}$



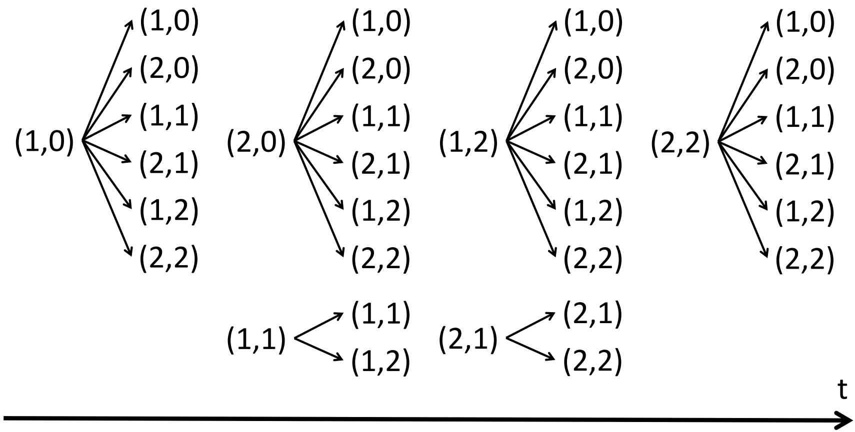
Regular schedule and expresses states sequences

Theorem 1.

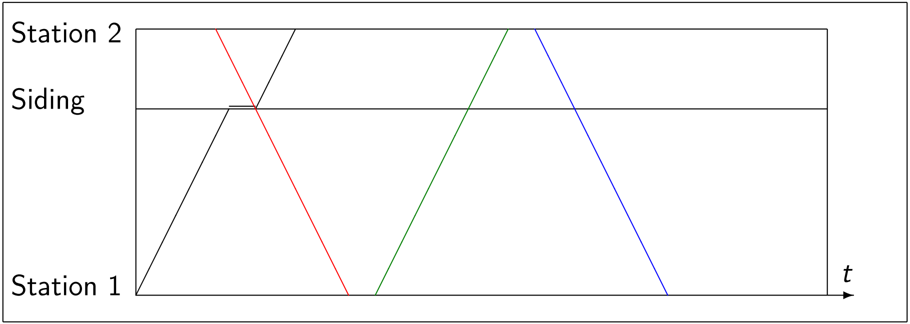
For each regular schedule there exists one and only one sequence of expresses states.



(2,1) (2,1) (2,2) (1,1) (1,1) (1,1) (1,2) (2,0)


(1,0)
(1,0)
(2,0)
(1,1)
(2,1)
(1,2)
(2,2)
(2,0)
(1,0)
(2,0)
(1,1)
(2,1)
(1,2)
(2,2)
(1,2)
(1,0)
(2,0)
(1,1)
(2,1)
(1,2)
(2,2)
(2,2)
(1,0)
(2,0)
(1,1)
(2,1)
(1,2)
(2,2)
(1,1)
(1,1)
(1,2)
(2,1)
(2,1)
(2,2)
t

States



$(2,2)$ $(1,0)$ $(2,0)$

Solution algorithm

Subproblem $\mathrm{P}(\boldsymbol{k_1}, \boldsymbol{k_2}, \boldsymbol{s}, \boldsymbol{b})$

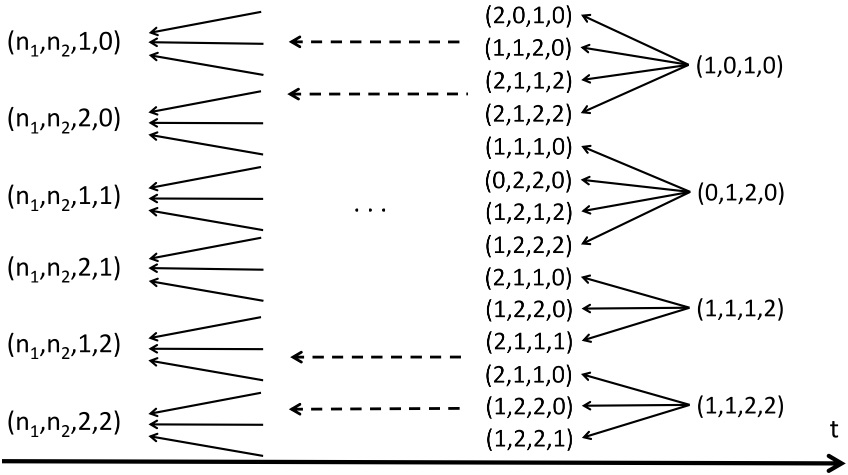
number of unsent trains
on station 1, $k_1 \in \{0,1,2,\dots,n_1\}$

additional condition:
state of the first express,
s$\in \{1,2\}$, $b \in \{0,1,2\}$

number of unsent trains on
station 2, $k_2 \in \{0,1,2,\dots,n_2\}$

Number of different subproblems – *$O(n^2)$*

Solution algorithm



Solution algorithm

Initial values

$$F(1,0,1,0)) = p_1 + p_2 - d_1^1;$$

$$F(0,1,2,0)) = p_1 + p_2 - d_2^1;$$

$$F(1,1,1,2) = \max\begin{cases} 2p_1 - d_2^1; \\ p_2 + p_1 - d_1^1; \end{cases}$$

$$F(1,1,2,2) = \max\begin{cases} 2p_2 - d_1^1; \\ p_2 + p_1 - d_2^1. \end{cases}$$

Exclusion of impossible subtasks

$F(0, k_2, 1, 0) = \infty;$

$F(k_1, 0, 2, 0) = \infty;$

$F(k_1, k_2, s, b) = \infty$ if $k_1 = 0$ or $k_2 = 0$, where $(s, b) \notin \{(1,0),(2,0)\}$.

Solution algorithm

Bellman equation

Optimal objective function value in the subproblem $P(k_1, k_2, s, b)$

$$F(k_1, k_2, s, b) = \min_{(k_1', k_2', s', b') \in T(k_1, k_2, s, b)} \max \begin{cases} H(k_1, k_2, s, b); \\ F(k_1', k_2', s', b') + g((s, b), (s', b')) \end{cases}$$

Objective function value of express in state (s, b) and skipping train

$$H(k_1, k_2, s, b) = \begin{cases} \max\{p_1 + p_2 - d_s^{k_s}; 2p_s - d_{\bar{s}}^{k_{\bar{s}}}\}, & \text{if } b = 2, \\ p_1 + p_2 - d_s^{k_s} & \text{otherwise.} \end{cases}$$

Solution algorithm

Algorithm for $\sum w_j C_j$

For objective function $\sum w_j C_j$ algorithm is the same, some operations and variables changes.

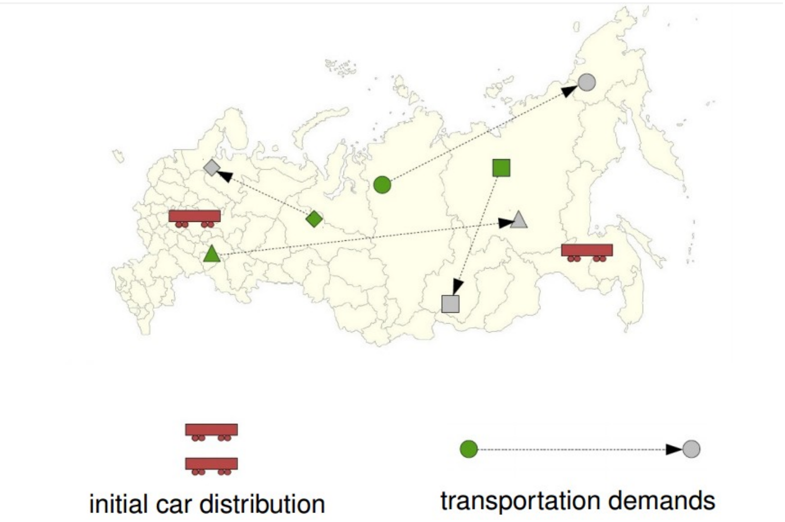
Single track railway scheduling problem with a siding

Results

Exact solution algorithm based on the dynamical programming method was proposed for the described problem.

Presented algorithm allows to construct set of optimal schedules in $O(n^2)$ operations.

The freight car routing problem: overview



Specificity of freight rail transportation in Russia

The state company

- Freight car blocking
- Freight train scheduling
- Locomotives management
- Personnel management

Transsp. costs matrix (M)
Transp. times matrix (D)

car movements

Independent freight car management companies

- Assignment of transportation demands to freight cars
- Freight car routing

Distances are large, and average freight train speed is low ($\approx$ 300 km/day): discretization in periods of 1 day is reasonable

The freight car routing problem: input and output

Input

Railroad network (stations)

Initial locations of cars (sources)

Transportation demands and associated profits

Costs: transfer costs and standing (waiting) daily rates;

Output: operational plan

A set of accepted demands and their execution dates

Empty and loaded cars movements to meet the demands (car routing)

Objective

Maximize the total net profit

Similar works in the literature

[Fukasawa, Poggi, Porto, Uchoa, ATMOS02]

- Train schedule is known
- Cars should be assigned to trains to be transported
- Discretization by the moments of arrival and departure of trains.
- Smaller time horizon (7 days)

Other works

- [Holmberg, Joborn, Lundren, TS98]
- [Löbel, MS98]
- [Campetella, Lulli, Pietropaoli, Ricciardi, ATMOS06]
- [Caprara, Malaguti, Toth, TS11]

Data: overview

T — planning horizon (set of time periods);

I — set of stations;

C — set of car types;

K — set of product types;

Q — set of demands;

S — set of sources (initial car locations);

M — empty transfer cost function;

D — empty transfer duration function;

Demands data

For each order $q \in Q$

- origin and destination stations;
- product type
- set of car types, which can be used for this demand — $C_q \subseteq C$
- maximum (minimum) number of cars, needed to fulfill (partially) the demand — $n_q^{\max}(n_q^{\min})$
- time window for starting the transportation
- profit vector (for delivery of one car with the product), depends on the period on which the transportation is started
- transportation time of the demand
- daily standing rates charged for one car waiting before loading (after unloading) the product at origin (destination) station

Sources and car types data

For each source $s \in S$

station where cars are located

type of cars

period, starting from which cars can be used

daily standing rate charged for cars

type of the latest delivered product

number of cars in the source — $\vec{n}_s \in \mathbb{N}$

For each car type $c \in C$

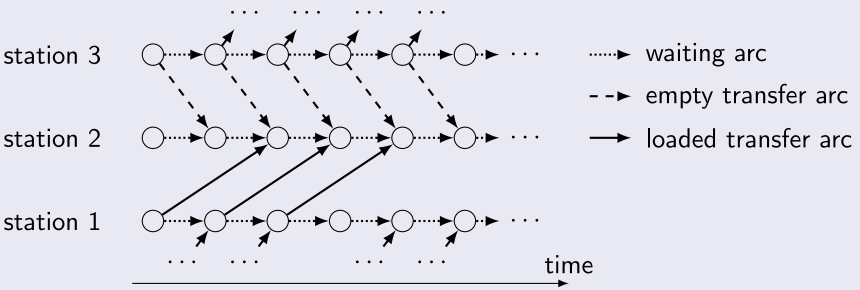
Q_c — set of demands, which a car of type c can fulfill

S_c — set of sources for car type c

Commodity graph

Commodity $c \in C$ represents the flow (movements) of cars of type c.

Graph $G_c = (V_c, A_c)$ for commodity $c \in C$:



Each vertex $v \in V_c$ represent location of cars of type c on a certain station at a certain time standing at a certain rate

g_a — cost of arc $a \in A_c$

Multi-commodity flow formulation

Variables

$x_a \in \mathbb{Z}_+$ — flow size along arc $a \in A_c$, $c \in C$

$y_q \in \{0,1\}$ — demand $q \in Q$ is accepted or not

$$\min \sum_{c \in C} \sum_{a \in A_c} g_a x_a$$

$$\sum_{c \in C_q} \sum_{a \in A_{cq}} x_a \le n_q^{\max} y_q \quad \forall q \in Q$$

$$\sum_{c \in C_q} \sum_{a \in A_{cq}} x_a \ge n_q^{\min} y_q \quad \forall q \in Q$$

$$\sum_{a \in \delta^-(v)} x_a - \sum_{a \in \delta^+(v)} x_a = \vec{n}_v \quad \forall c \in C, v \in V_c$$

$$x_a \in \mathbb{Z}_+ \quad \forall c \in C, a \in V_c$$

$$y_q \in \{0,1\} \quad \forall q \in Q$$

We concentrate on solving its LP-relaxation

Multi-commodity flow formulation

Variables

$x_a \in \mathbb{Z}_+$ — flow size along arc $a \in A_c$, $c \in C$

$y_q \in \{0,1\}$ — demand $q \in Q$ is accepted or not

$$\min \sum_{c \in C} \sum_{a \in A_c} g_a x_a$$

$$\sum_{c \in C_q} \sum_{a \in A_{cq}} x_a \le n_q^{\max} y_q \qquad \forall q \in Q$$

$$\sum_{c \in C_q} \sum_{a \in A_{cq}} x_a \ge n_q^{\min} y_q \qquad \forall q \in Q$$

$$\sum_{a \in \delta^-(v)} x_a - \sum_{a \in \delta^+(v)} x_a = \vec{n}_v \qquad \forall c \in C, v \in V_c$$

$$x_a \in \mathbb{Z}_+ \qquad \forall c \in C, a \in V_c$$

$$y_q \in \{0,1\} \qquad \forall q \in Q$$

We concentrate on solving its LP-relaxation

Multi-commodity flow formulation

Variables

$x_a \in \mathbb{Z}_+$ — flow size along arc $a \in A_c$, $c \in C$

$y_q \in \{0, 1\}$ — demand $q \in Q$ is accepted or not

$$\min \sum_{c \in C} \sum_{a \in A_c} g_a x_a$$

$$\sum_{c \in C_q} \sum_{a \in A_{cq}} x_a \leq n_q^{\max} y_q \qquad \forall q \in Q$$

$$\sum_{c \in C_q} \sum_{a \in A_{cq}} x_a \geq n_q^{\min} y_q \qquad \forall q \in Q$$

$$\sum_{a \in \delta^-(v)} x_a - \sum_{a \in \delta^+(v)} x_a = \vec{n}_v \qquad \forall c \in C, v \in V_c$$

$$0 \leq x_a \qquad \forall c \in C, a \in V_c$$

$$0 \leq y_q \leq 1 \qquad \forall q \in Q$$

We concentrate on solving its LP-relaxation

Path reformulation

P_s — set of paths (car routes) from source $s \in S$

Variables

$\lambda_s \in \mathbb{Z}_+$ — flow size along path $p \in P_s$, $s \in S$

$$\min \sum_{c \in C} \sum_{s \in S_c} \sum_{p \in P_s} g_p^{path} \lambda_p$$

$$\sum_{c \in C_q} \sum_{s \in S_c} \sum_{p \in P_s:\ q \in Q_p^{path}} \lambda_a \leq n_q^{\max} y_q \quad \forall q \in Q$$

$$\sum_{c \in C_q} \sum_{s \in S_c} \sum_{p \in P_s:\ q \in Q_p^{path}} \lambda_a \geq n_q^{\min} y_q \quad \forall q \in Q$$

$$\sum_{p \in P_s} \lambda_p = \vec{n}_s \quad \forall c \in C, s \in S_c$$

$$\lambda_p \in \mathbb{Z}_+ \quad \forall c \in C, s \in S_c, p \in P_s$$

$$y_q \in \{0, 1\} \quad \forall q \in Q$$

Column generation for path reformulation

Pricing problem decomposes to shortest path problems, one for each source

- slow: number of sources are thousands

Column generation for path reformulation

Pricing problem decomposes to shortest path problems, one for each source

> slow: number of sources are thousands

To accelerate, for each commodity $c \in C$, we search for a shortest path in-tree to the terminal vertex from all sources in S_c

> drawback: some demands are severely "overcovered", bad convergence

Column generation for path reformulation

Pricing problem decomposes to shortest path problems, one for each source

> slow: number of sources are thousands

To accelerate, for each commodity $c \in C$, we search for a shortest path in-tree to the terminal vertex from all sources in S_c

> drawback: some demands are severely "overcovered", bad convergence

We developed iterative procedure which removes covered demands and cars assigned to them, and the repeats search for a shortest path in-tree

Flow enumeration reformulation

F_c — set of fixed flows for commodity $c \in C$

Variables

$\omega_f \in \{0, 1\}$ — commodity c is routed accordity to flow $f \in F_c$ or not

$$\min \sum_{c \in C} \sum_{f \in F_s} g_f^{flow} \omega_f$$

$$\sum_{c \in C_q} \sum_{f \in F_c} \sum_{a \in A_{cq}} f_a \omega_f \le n_q^{\max} y_q \quad \forall q \in Q$$

$$\sum_{c \in C_q} \sum_{f \in F_c} \sum_{a \in A_{cq}} f_a \omega_f \ge n_q^{\min} y_q \quad \forall q \in Q$$

$$\sum_{f \in F_c} \omega_f = 1 \quad \forall c \in C$$

$$\omega_p \in \{0, 1\} \quad \forall c \in C, f \in F_c$$

$$y_q \in \{0, 1\} \quad \forall q \in Q$$

Approach CGEF

Pricing problem decomposes to minimum cost flow problems, one for each commodity

slow: very bad convergence

<+-> If an arc flow variable x has a negative reduced cost, there exists a negative reduced cost pricing problem solution in which $x > 0$. (consequence of the theorem in [S. and Vanderbeck, 13])

Approach CGEF

Pricing problem decomposes to minimum cost flow problems, one for each commodity

slow: very bad convergence

"Column generation for extended formulations" (CGEF) approach: we disaggregate the pricing problem solution to arc flow variables, which are added to the master.

<+-> If an arc flow variable x has a negative reduced cost, there exists a negative reduced cost pricing problem solution in which $x > 0$.
(consequence of the theorem in [S. and Vanderbeck, 13])

Approach CGEF

Pricing problem decomposes to minimum cost flow problems, one for each commodity

slow: very bad convergence

"Column generation for extended formulations" (CGEF) approach: we disaggregate the pricing problem solution to arc flow variables, which are added to the master.

The master then becomes the multi-commodity flow formulation with restricter number of arc flow variables, i.e. "improving" variables are generated dynamically

<+-> If an arc flow variable x has a negative reduced cost, there exists a negative reduced cost pricing problem solution in which $x > 0$. (consequence of the theorem in [S. and Vanderbeck, 13])

Tested approaches

DIRECT: solution of the multi-commodity flow formulation by the *Clp* LP solver

- Problem specific solver source code modifications
- Problem specific preprocessing is applied (not public)
- Tested inside the company

Tested approaches

DIRECT: solution of the multi-commodity flow formulation by the *Clp* LP solver

- Problem specific solver source code modifications
- Problem specific preprocessing is applied (not public)
- Tested inside the company

COLGEN: solution of the path reformulation by column generation (*BaPCod* library and *Cplex* LP solver)

- Initialization of the master by “doing nothing” routes
- Stabilization by dual prices smoothing
- Restricted master clean-up

Tested approaches

DIRECT: solution of the multi-commodity flow formulation by the *Clp* LP solver

- Problem specific solver source code modifications
- Problem specific preprocessing is applied (not public)
- Tested inside the company

COLGEN: solution of the path reformulation by column generation (*BaPCod* library and *Cplex* LP solver)

- Initialization of the master by "doing nothing" routes
- Stabilization by dual prices smoothing
- Restricted master clean-up

COLGENEF: "dynamic" solution of multi-commodity flow formulation by the CGEF approach (*BaPCod* library, *Lemon* min-cost flow solver and *Cplex* LP solver)

- Initialization of the master by all waiting arcs
- Only trivial preprocessing is applied

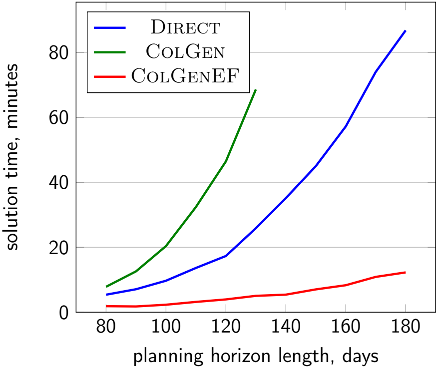
First test set of real-life instances

Instance name	x3	x3double	5k0711q
Number of stations	371	371	1'900
Number of demands	1'684	3'368	7'424
Number of car types	17	17	1
Number of cars	1'013	1'013	15'008
Number of sources	791	791	11'215
Time horizon, days	37	74	35
Number of vertices, thousands	62	152	22
Number of arcs, thousands	794	2'846	1'843
Solution time for DIRECT	20s	1h34m	55s
Solution time for COLGEN	22s	7m53s	8m59s
Solution time for COLGENEF	3m55s	>2h	43s

Real-life instances with larger planning horizon

1'025 stations, up to 6'800 demands, 11 car types, 12'651 cars, and 8'232 sources.

Up to $\approx$ 300 thousands nodes and 10 millions arcs.



Horizon	DIRECT	COLGENEF
80	5m24s	1m52s
90	7m05s	1m47s
100	9m42s	2m19s
110	13m38s	3m11s
120	17m19s	3m57s
130	25m52s	5m03s
140	35m08s	5m25s
150	44m58s	7m02s
160	57m11s	8m19s
170	1h13m58s	10m53s
180	1h26m46s	12m16s

Convergence of COLGENEF in less than 15 iterations.

About 3% of arc flow variables at the last iteration.

Conclusions

- Three approaches tested for a freight car routing problem on real-life instances
- Approach ColGen is the best for instances with small number of sources
- Problem-specific preprocessing is important: good results for Direct
- Approach ColGenEF is the best for large instances
- Combination of ColGenEF and problem-specific preprocessing would allow to increase discretization and improve solutions quality

Perspectives

Some practical considerations are not taken into account:

- Progressive standing daily rates
- Special stations for long-time stay (with lower rates)
- Compatibility between two consecutive types of loaded products.
- Penalties for refused demands
- Groups of cars are transferred faster and for lower unitary costs.

RCPSP

Resource Constrained Project Scheduling Problem (RCPSP)

Considers resources of limited availability and activities of known durations and resource requests, linked by precedence relations. The problem consists of finding a schedule of minimal duration by assigning a start time to each activity such that the precedence relations and the resource availabilities are respected.

Examples of RCPSP

Plannig of production and maintenance processes on the enterprise.

Software development tasks distribution.

Planning of training processes.

Number of publications in last 5 years

Keyword	GoogleScholar	Science Direct
RCPSP	1 560	161
project scheduling	73 300	63 694

Classical RCPSP formulation

Set of renewable resources R

c_i – capacity of resource $X_i \in R$.

Set of activities $N = \{A_1, \dots, A_n\}$

$|N| = n$;

$G(N, E)$ – precedence relations graph;

r_j – release time of $A_j \in N$;

p_j – processing time of $A_j \in N$;

a_{ji} – amount of resource $X_i \in R$ required to process $A_j \in N$.

All variables belong to Z_+.

Classical RCPSP formulation

Schedule π

$S_j(\pi)$ – start time of activitiy $A_j \in N$ under π;

$C_j(\pi) = S_j(\pi) + p_j$ – completion time of task $A_j \in N$ under π.

Feasible schedules $\Pi(N, R)$

$S_j(\pi) \geq r_j$ holds for any $A_j \in N, \pi \in \Pi(N, R)$ – release times not violated;

$C_j(\pi) \leq S_k(\pi)$ for any $e_{jk} \in E$ – precedence relations satisfied;

$\sum\limits_{j \in N: S_j(\pi) \leq t < C_j(\pi)} a_{ji} \leq c_i$ for any $X_i \in R,\ t \geq 0$ – resource capacity not violated.

Classical RCPSP formulation

Problem statement

The RCPSP is the problem of finding a feasible schedule of minimal makespan subject to precedence constraints and resource constraints, i.e.

$$\min_{\pi \in \Pi(N,R)} \max_{A_j \in N} C_j(\pi).$$

Complexity

Problem is NP-complete in a strong sense (Garey, Johnson 1975).

Example

Problem data

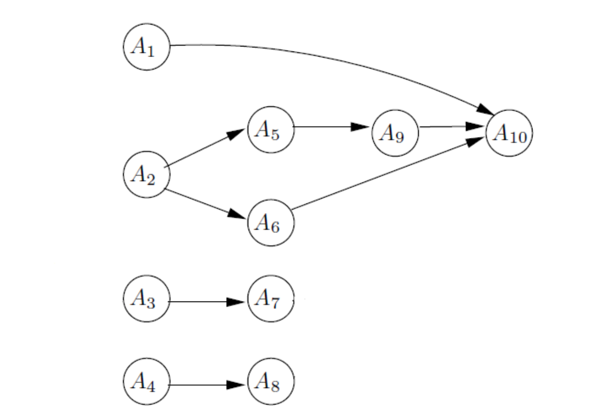
2 resources X_1 and X_2 with capacities $c_1 = 7$ and $c_2 = 4$;

10 activities.

A_j	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}
p_j	6	1	1	2	3	5	6	3	2	4
a_{j1}	2	1	3	2	1	2	3	1	1	1
a_{j2}	1	0	1	0	1	1	0	2	2	1

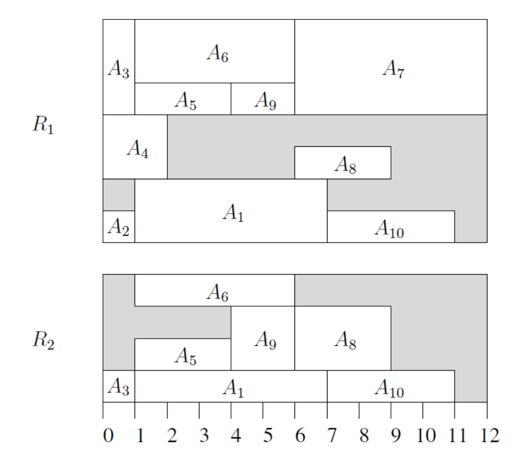
Example

Precedence relations



Example

Schedule with minimal makespan



RCPSP

Decision variant of RCPSP

The decision variant of the RCPSP is the problem of determining whether a schedule π of makespan not greater than H subject to precedence and resource constraints exists or not.

NP-complete in a strong sense

Garey and Johnson (1975) have shown that the decision variant of the RCPSP with a single resource and no precedence constraints, called the resource-constrained scheduling problem, is NP-complete in the strong sense by reduction from the 3-partition problem.

Exact solution methods for RCPSP

There is a variety of methods to find the exact solutions. Most of them are based on the following ideas.

- Branch-and-Bound approach;
- Column Generation;
- Constraint Programming.

Makespan lower bound

Correct makespan lower bound

LB – amount of time which is not higher than makespan value for any schedule $\pi \in \Pi(N, R)$, i.e.

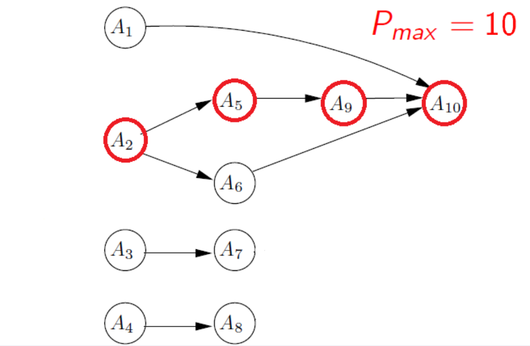
$$LB \leq \max_{A_j \in N} C_j(\pi).$$

Existed lower bound estimation methods

Critical path

P_{max} – length of the longest path in graph $G(N, E)$.
Makespan is not lower than critical path length for any $\pi \in \Pi(N, R)$.

$$P_{\max} = LB_P.$$



Existed lower bound estimation methods

Resource load

$RL_i = \sum_{A_j \in N} p_j a_{ji}$ – total amount of reource X_i required for the project.

Then, under any feasible schedule makespan value should be enough to use requred amount of any resource $X_i \in R$ subject to its capacity, i.e.

$$LB_R = \lceil \max_{i \in R} \frac{RL_i}{c_i} \rceil.$$

In our example

$$\frac{RL_1}{c_1} = \frac{60}{7} = 8\frac{4}{7}, \ \frac{RL_2}{c_2} = \frac{29}{4} = 7\frac{1}{4},$$

$$LB_R = \lceil 8\frac{4}{7} \rceil = 9.$$

Destructive lower bound techniques

Deals with decision variant of RCPSP. The objective is to prove that for defined horizon H there are no feasible schedule with makespan not higher than H:

- desjunctive lower bounds i.e. maximum clique computation;
- Linear Programming (LP) relaxations;
- relaxations of decision variant of RCPSP to Cumulative Scheduling Problem (CuSP);
- other constraint programming based approaches;
- exact methods of solving decision variant of RCPSP.

Existed lower bound estmiation methods

Satisfiability tests (SAT)

1. Find makespan lower bound LB and upper bound UB using algorithms with low computational complexity.
2. Consider time horizon H such as $LB \leq H \leq UB$ and use some of *destructive lower bound* techniques to check the existance of feasible schedule with makespan not lower than H.
3. Use logarithmic search to find the highest horizon H^* which not allows the existance of feasible schedule.
4. Set the lower bound equals to $H^* + 1$.

Existed lower bound estmiation methods

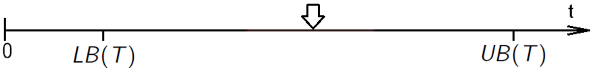
Satisfiability tests (SAT)

1. Find makespan lower bound LB and upper bound UB using algorithms with low computational complexity.
2. Consider time horizon H such as $LB \leq H \leq UB$ and use some of *destructive lower bound* techniques to check the existance of feasible schedule with makespan not lower than H.
3. Use logarithmic search to find the highest horizon H^* which not allows the existance of feasible schedule.
4. Set the lower bound equals to $H^* + 1$.

Existed lower bound estmiation methods

Satisfiability tests (SAT)

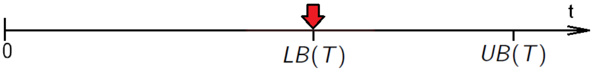
1. Find makespan lower bound LB and upper bound UB using algorithms with low computational complexity.
2. Consider time horizon H such as $LB \leq H \leq UB$ and use some of *destructive lower bound* techniques to check the existance of feasible schedule with makespan not lower than H.
3. Use logarithmic search to find the highest horizon H^* which not allows the existance of feasible schedule.
4. Set the lower bound equals to $H^* + 1$.



Existed lower bound estmiation methods

Satisfiability tests (SAT)

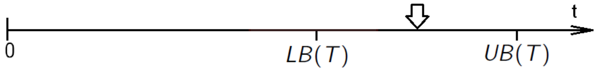
1. Find makespan lower bound LB and upper bound UB using algorithms with low computational complexity.
2. Consider time horizon H such as $LB \leq H \leq UB$ and use some of *destructive lower bound* techniques to check the existance of feasible schedule with makespan not lower than H.
3. Use logarithmic search to find the highest horizon H^* which not allows the existance of feasible schedule.
4. Set the lower bound equals to $H^* + 1$.



Existed lower bound estmiation methods

Satisfiability tests (SAT)

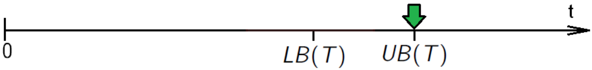
1. Find makespan lower bound LB and upper bound UB using algorithms with low computational complexity.
2. Consider time horizon H such as $LB \leq H \leq UB$ and use some of *destructive lower bound* techniques to check the existance of feasible schedule with makespan not lower than H.
3. Use logarithmic search to find the highest horizon H^* which not allows the existance of feasible schedule.
4. Set the lower bound equals to $H^* + 1$.



Existed lower bound estmiation methods

Satisfiability tests (SAT)

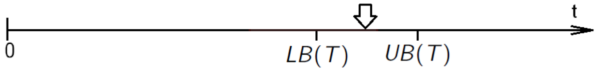
1. Find makespan lower bound LB and upper bound UB using algorithms with low computational complexity.
2. Consider time horizon H such as $LB \leq H \leq UB$ and use some of *destructive lower bound* techniques to check the existance of feasible schedule with makespan not lower than H.
3. Use logarithmic search to find the highest horizon H^* which not allows the existance of feasible schedule.
4. Set the lower bound equals to $H^* + 1$.



Existed lower bound estmiation methods

Satisfiability tests (SAT)

1. Find makespan lower bound LB and upper bound UB using algorithms with low computational complexity.
2. Consider time horizon H such as $LB \leq H \leq UB$ and use some of *destructive lower bound* techniques to check the existance of feasible schedule with makespan not lower than H.
3. Use logarithmic search to find the highest horizon H^* which not allows the existance of feasible schedule.
4. Set the lower bound equals to $H^* + 1$.



Existed lower bound estimation methods

Constraint Propagation to tighten the problem

These approaches makes an interval $[r_j, D_j]$ of possible processing of activity $A_j \in N$ tighter and improve the performances of algorithms. The most popular approaches are:

- *timetabling* techniques are based on the computation of an aggregation of the resource demand at every time-point;
- *edge finding* and *activity intervals* techniques rely on the analysis of the resource demand over time intervals;
- *conjunctive reasoning with temporal constraints* are based on an analysis of the current temporal constraint network.

Existed lower bound estmiation methods

Trivial algorithms

Advantages: low calculation complexity, algorithms can be applied for large-scaled problems.
Disadvantages: low precision of obtained bound.

Advanced algorithms

Advantages: high precision of obtained bound.
Disadvantages: exponential complexity decrease the efficiency of obtained bound and make algorithms not possible to be applied for some large-scaled problems.

Problem!

There is a strong need in the method which can obtain suitable lower bound for large-scaled instances!

RCPSP

Some generalizations of RCPSP

RCPSP with time-dependent resource capacities.

RCPSP with minimal and maximal time lags (RCPSP/max) – generalized precedence relations express relations of start-to-start, start-to-end, end-to-start, and end-to-end times between pairs of activities.

Multi-Mode RCPSP (MRCPSP) – activities can be processed in several modes each of which charachterized by processing time and required amounts of resources.

RCPSP with flexible resource profile (FRCPSP) – only total amounts of required resources are given for activies instead, processing times are not defined.

RCPSP

PSPLIB benchmark

The library of instances of problems RCPSP, RCPSP/max, MRCPSP, MRCPSP/max, FRCPSP and others.
Website: http://www.om-db.wi.tum.de/psplib/main.html

Kolisch, R. and A. Sprecher (1996)

PSPLIB - A project scheduling library // *European Journal of Operational Research*, Vol. 96, pp. 205–216.

R. Kolisch, C. Schwindt und A.Sprecher (1999)

Benchmark instances for project scheduling problems *In: Kluwer; Weglarz, J. (Hrsg.): Handbook on recent advances in project scheduling*, pp. 197-212.

Our publications

Лазарев А.А., Бронников С.В., Герасимов А.Р., Мусатова Е.Г., Петров А.С., Пономарев К.В., Харламов М.М., Хуснуллин Н.Ф., Ядренцев Д.А. Математическое моделирование планирования подготовки космонавтов // Управление большими системами, 2016 (принято к печати)

Musatova E., Lazarev A., Ponomarev K., Yadrentsev D., Bronnikov S., Khusnullin N. A Mathematical Model for the Astronaut Training Scheduling Problem // IFAC-PapersOnLine, Volume 49, Issue 12, 2016, Pages 221-225.

Бронников С.В., Долгий А.Б., Лазарев А.А., Морозов Н.Ю., Петров А.С., Садыков Р.Р., Сологуб А.А., Вернер Ф., Ядренцев Д.А., Мусатова Е.Г., Хуснуллин Н.Ф. Approaches for planning the ISS cosmonaut training. Preprint Nr. 12. Magdeburg: Institut fur Mathematische Optimierung, 2015. – 33 с.

С.В.Бронников, А.Р.Герасимов, А.А.Лазарев, Е.Г.Мусатова, А.С.Петров, К.В.Пономарев, М.М.Харламов, Н.Ф.Хуснуллин, Д.А.Ядренцев. К решению задачи автоматизации планирования подготовки космонавтов для работы на МКС / Труды 7-й Международной научной конференции «Теория расписаний и методы декомпозиции. Танаевские чтения» (Беларусь, Минск, 2016). Минск: ОИПИ НАН Беларуси, 2016. С. 23-27.

Our publications

Бронников С.В., Лазарев А.А., Морозов Н.Ю., Харламов М.М., Ядренцев Д.А. Mathematical models and approaches in problem of volume planning of ISS cosmonauts trainings / Abstracts of the 28th Conference of the European Chapter on Combinatorial Optimization (Catania, 2015), 2015. С. 61.

Бронников С.В., Лазарев А.А., Петров А.С., Ядренцев Д.А. Models and Approaches for Planning the ISS Cosmonaut Training / Труды VI International Conference on Optimization Methods and Applications. (OPTIMA-2015, Petrovac). М.: ФГБУН ВЦ им. А.А.Дородницына РАН, 2015. С. 196-197 http://www.cima.uevora.pt/optima2015/Optima2015.pdf.

Лазарев А.А., Петров А.С., Сологуб А.А., Гущина В.П., Морозов Н.Ю. Модели и алгоритмы решения задач объёмно-календарного планирования подготовки экипажа МКС / Тезисы докладов Всероссийской молодёжной научно-практической конференции «Космодром «Восточный» и перспективы развития российской космонавтики» (Благовещенск, 2015). Благовещенск: СГАУ, 2015. С. 200-201.

Ядренцев Д.А., Бронников С.В., Лазарев А.А., Мусатова Е.Г., Хуснуллин Н.Ф. Календарное планирование подготовки космонавтов к выполнению космического полета / Материалы 11-й Международной научно-практической конференции "Пилотируемые полеты в космос"(Звездный городок, 2015). Звездный городок: ФБГУ ЦПК им. Ю.А. Гагарина», 2015. С. 95-96.

Open problems

1 machine

2 jobs

r_j — release times

p_j — processing times

d_j — due dates

…and so on — we assume that all parameters of the jobs are known beforehand.

k known schedules $\pi_{-k}, \pi_{-k+1}, \ldots, \pi_{-1}$ that are optimal according to an unknown objective function

The goal is to construct a new schedule π that would be optimal according to the same objective function, or at least would approximate the optimal. Perhaps the discrepancy between obtained solution and the optimal schedule would decrease with the number of known schedules?

Scheduling Theory and Applications

Alexander Lazarev

Lomonosov Moscow State University

National Research University Higher School of Economics

Moscow Institute of Physics and Technology (State University)

V.A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences (ICS RAS)

jobmath@mail.ru

www.orsot.ru

Thank you for your attention!