

# Scheduling Theory and Applications. Part I

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

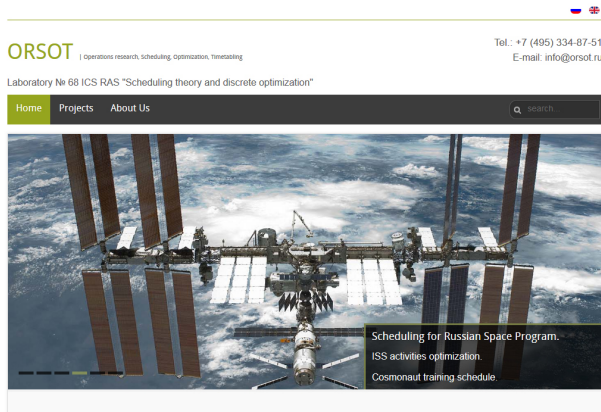
www.orsot.ru



V.A. TRAPEZNIKOV  
INSTITUTE  
OF CONTROL  
SCIENCES  
OF RUSSIAN ACADEMY  
OF SCIENCES



- 1 About ORSOT
  - Laboratory №68
  - Projects
- 2 History of Scheduling Theory
  - Gantt chart
  - Scheduling theory term
- 3 Pioneers of scheduling theory
  - J. R. Jackson. Scheduling a production to minimize maximum tardiness.
  - W. E. Smith. Various optimizers for single-stage production.
  - S. M. Johnson. Optimal two-and-three-stage production schedules with set-up times included.
- 4 Problem of two production lines



ORSOT | Operations research, Scheduling, Optimization, Timetabling

Tel.: +7 (495) 334-87-51  
E-mail: info@orsot.ru

Laboratory № 68 ICS RAS "Scheduling theory and discrete optimization"

Home Projects About Us

search

Scheduling for Russian Space Program.  
ISS activities optimization.  
Cosmonaut training schedule.

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](http://orsot.ru) (Operation Research Scheduling Optimization Timetabling)



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





Transport logistics  
Forming trains and routes



Optimizing assembly lines  
Balancing and rebalancing assembly lines  
Distributing operations



Composing study schedules  
Program product on 1C platform

# Our team

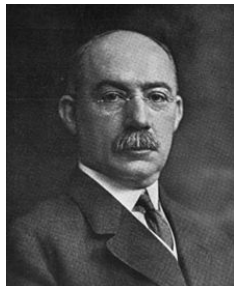
Members of our laboratory teach in Lomonosov Moscow State University, National Research University Higher School of Economics and Moscow Institute of Physics and Technology. A number of members is seeking D.Sc. and Ph.D. degrees in Russia, France and Germany.

- Prof. D.Sc. Alexander Lazarev, Head of Laboratory
- Prof. D. Sc. Mikhail Ulyanov, Senior Researcher
- D.Sc. Evgeny Gafarov, Senior Researcher
- Elena Musatova, Ph.D., Senior Researcher
- Ivan Nekrasov, Ph.D., Senior Researcher
- Nail Khusnullin, Senior Engineer and Software Developer
- Dmitry Arkhipov, Senior Engineer
- Alexei Gerasimov, Senior Engineer
- Irina Golubeva, Senior Engineer
- Elina Karnysheva, Senior Engineer
- Nikolay Loginov, Middle Engineer
- Nikolay Pravdivets, Middle Engineer
- Alexey Petrov, Middle Engineer
- Ilya Tarasov, Middle Engineer
- German Tarasov, Junior Engineer

- Prof. Alexandre Dolgui, IMT Atlantique (former Ecole des Mines de Nantes), France
- Apl. Prof. Frank Werner, Otto-von-Guericke-University Magdeburg, Germany
- Prof. Bertrand M.T. Lin, National Chiao Tung University, Taiwan
- Prof. Edwin Cheng, Hong Kong Polytechnic University, Hong Kong
- Ruslan Sadykov, Institut de Mathématique de Bordeaux, France
- Prof. Yakov Zinder, University of Technology Sydney, Australia
- Prof. Dr. Erwin Pesch, University of Siegen, Germany

- 1C COMPANY
- Russian Railways (RZD)
- Yu. A. Gagarin Research&Test Cosmonaut Training Center (GCTC)
- State Space Corporation ROSCOSMOS
- Smart Solutions

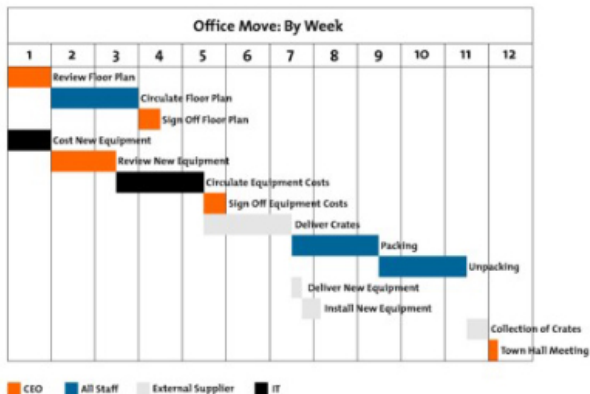
# Gantt chart



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

# 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

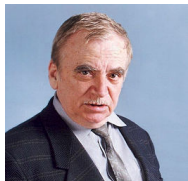
## First monograph on Scheduling Theory

R. W. Conway, W. L. Maxwell, L. W. Miller. *Theory of Scheduling*, 1967  
(Russian edition in 1975)

# Pioneers of scheduling theory in USSR



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



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

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

# 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.

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

## Restrictions:

- 1 Processing of a job  $j \in N$  cannot be started earlier than the release time of this job.
- 2 The machine can only process one job at a time.
- 3 No interruptions are allowed (non-preemptive scheduling): once processing of some job has started, it should continue until fully completed.
- 4 Due dates  $d_j$  can be violated (although it's undesirable).  
Deadlines  $D_j$  must not be violated at all. It's assumed that this condition can be fulfilled, i. e.  $r_j + p_j \leq D_j$ .

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

**Note:** there are two most common ways of schedule representation:

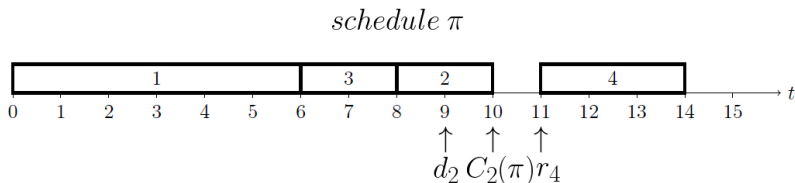
- The schedule is represented by a permutation of jobs (in what order the jobs should be processed), e. g.,  $\pi = (6, 3, 2, 1, \dots)$  means that at first job 6 is processed, then job 3, then 2 and so on.
- The schedule is represented by a vector of processing start times  $S_j$  or vector of processing completion times  $C_j$ . For example,  $\pi = (10, 0, 11, 5, 6, 4, \dots)$  (in case of start times) means that job 1 starts at  $t = 10$ , job 2 starts at  $t = 0$ , 3 starts at  $t = 11$  and so on.

Depending on formulation of the problem, one method or another may be more "convenient" to implement. In some cases (e. g. in Jackson's problem) these methods are equivalent and can be easily converted one into another.

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

Schedule  $\pi$  (permutation of jobs or vector of completion times)

$C_j(\pi)$  — completion time of job  $j$



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

Later in the 20th century, after Jackson's article was completed and published, the terminology of scheduling theory had changed. What we now call *lateness* before this change was called *tardiness*, so, using modern terminology, Jackson was considering exactly lateness, and the title of his article is outdated.

Note: "Lateness" is any deviation from the due date  $d_j$ . Positive lateness is "tardiness", negative lateness is "earliness".

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

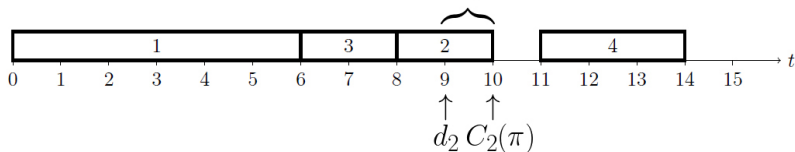
*Tardiness* of job  $j$ :  $\max\{C_j(\pi) - d_j, 0\}$

*Earliness* of job  $j$ :  $\max\{d_j - C_j(\pi), 0\}$ .

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

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

Lateness of job 2:  $C_2(\pi) - d_2 = 1$



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

Schedule  $\pi$  (permutation of jobs or vector of completion times)

$C_j(\pi)$  — completion time of job  $j$

The goal is to construct a schedule with minimal value of maximum lateness. Maximum lateness in this case is the objective function of this problem:

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

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

Schedule  $\pi$  (permutation of jobs or vector of completion times)

$C_j(\pi)$  — completion time of job  $j$

The goal is to construct a schedule with minimal value of maximum lateness. Maximum lateness in this case is the objective function of this problem:

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

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

Schedule  $\pi$  (permutation of jobs or vector of completion times)

$C_j(\pi)$  — completion time of job  $j$

The goal is to construct a schedule with minimal value of maximum lateness. Maximum lateness in this case is the objective function of this problem:

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

**Try to answer these questions:** What is the purpose of this objective function? Why is this objective function exactly the way it is? Does it possibly reflect some traits of human behavior or is it just a mathematical construction?

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

How you would solve this problem?

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

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  $\pi^* = (j_1, j_2, \dots, j_n)$  includes jobs that are sorted according to non-decrease of their due dates:

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

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

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  $\pi^* = (j_1, j_2, \dots, j_n)$  includes jobs that are sorted according to non-decrease of their due dates:

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

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.

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  $\pi^* = (j_1, j_2, \dots, j_n)$  includes jobs that are sorted according to non-decrease of their due dates:

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

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

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

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

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  $\pi^* = (j_1, j_2, \dots, j_n)$  includes jobs that are sorted according to non-decrease of their due dates:

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

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

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

What if this requirement is not fulfilled? (i. e.  $\exists j \in N : r_j \neq 0$ )

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

1 machine

1 machine

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

1 machine

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

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

$p_j$  — processing time

## Restrictions:

- 1 Processing of a job cannot be started earlier than the release time of this job.
- 2 Only one job may be processed simultaneously by the machine.
- 3 No interruptions are allowed (non-preemptive scheduling): once processing of some job has started, it should continue until fully completed.

Schedule  $\pi$  (permutation if jobs)

Schedule  $\pi$  (permutation if jobs)

$C_j(\pi)$  — completion time of job  $j$

Schedule  $\pi$  (permutation of jobs)

$C_j(\pi)$  — completion time of job  $j$

Objective function: minimum total completion time

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

Schedule  $\pi$  (permutation of jobs)

$C_j(\pi)$  — completion time of job  $j$

Objective function: minimum total completion time

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

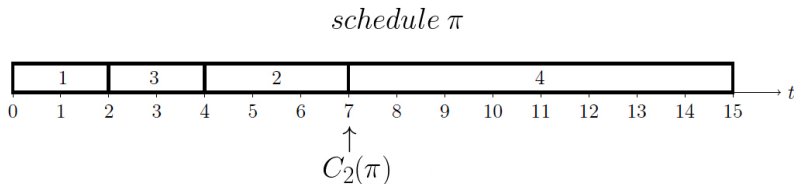
**Try to answer these questions:** What is the purpose of this objective function? Why is this objective function exactly the way it is? Does it possibly reflect some traits of human behavior or is it just a mathematical construction?

How you would solve this problem?

How you would solve this problem?

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

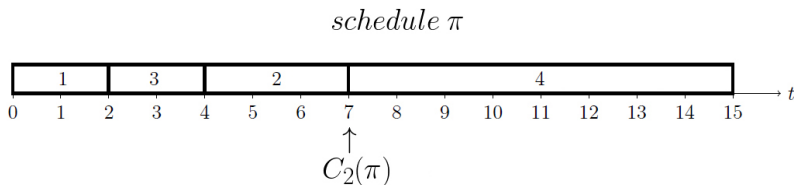
$$p_{j_1} \leq p_{j_2} \leq \dots \leq p_{j_n}$$



How you would solve this problem?

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

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



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

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

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

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

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

# 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.

## Restrictions:

- 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  $t + p_i^j$

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

Schedule  $\pi$  (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.

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

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

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

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

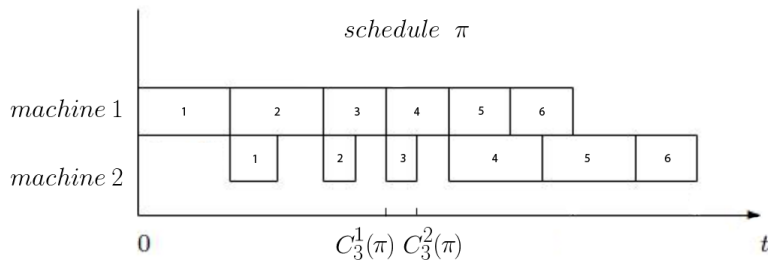
$C_j^1(\pi), C_j^2(\pi)$  — completion times on machines 1 and 2 respectively

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:



# Johnson's algorithm:

Algorithm 1. Input: set  $N_{12}$  of jobs. Output: permutation  $\pi$ .

**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

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

# Johnson's algorithm:

Algorithm 1. Input: set  $N_{12}$  of jobs. Output: permutation  $\pi$ .

**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

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

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

# Johnson's algorithm:

Algorithm 1. Input: set  $N_{12}$  of jobs. Output: permutation  $\pi$ .

**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

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

**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  $\pi_1$ :  $\pi_1 := \pi_1 \cup i_1$ , otherwise, if  $p_{i_1} = p_{i_1}^2$ , add it to  $\pi_2$ :  $\pi_2 := i_1 \cup \pi_2$

# Johnson's algorithm:

Algorithm 1. Input: set  $N_{12}$  of jobs. Output: permutation  $\pi$ .

**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

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

**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  $\pi_1$ :  $\pi_1 := \pi_1 \cup i_1$ , otherwise, if  $p_{i_1} = p_{i_1}^2$ , add it to  $\pi_2$ :  $\pi_2 := i_1 \cup \pi_2$

**Step 4**  $\pi := \pi_1 \cup \pi_2$

**End.**

# Johnson's algorithm:

Algorithm 2. Input: permutation of jobs  $\pi$ . 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  $\pi$ . 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.

# Johnson's algorithm:

Algorithm 2. Input: permutation of jobs  $\pi$ . 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  $\pi$ . 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:  $C_j^i = S_j^i + p_j^i$ ,  $i = 1, 2$ ,

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

$$S_{j_1}^1 = 0, \quad S_{j_k}^1 = C_{j_{k-1}}^1, \quad k \geq 2$$

$$S_{j_1}^2 = c_{j_1}^1, \quad S_{j_k}^2 = \max\{C_{j_k}^1, C_{j_{k-1}}^2\}, \quad k \geq 2$$

# 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

$$\pi = ( \quad , \quad , \quad , \quad , \quad , \quad , \quad )$$

$$\pi_1 = ( \quad )$$

$$\pi_2 = ( \quad )$$

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

$$\pi = (4, \quad , \quad , \quad , \quad , \quad , \quad )$$

$$\pi_1 = (4)$$

$$\pi_2 = ()$$

# 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

$$\pi = (4, 3, \quad, \quad, \quad, 2, 5)$$

$$\pi_1 = (4, 3)$$

$$\pi_2 = (2, 5)$$

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

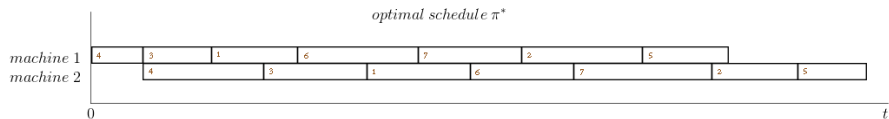
$p_j^1$    7    6

$p_j^2$    6    8

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

$\pi = \pi_1 \cup \pi_2$

$O(n \log n)$



# 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  $\rightarrow$  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 " $2 \rightarrow 1$ " and " $1 \rightarrow 1$ " and " $2 \rightarrow 2$ "? (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 " $2 \rightarrow 1$ " and " $1 \rightarrow 1$ " and " $2 \rightarrow 2$ "? (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  $\rightarrow$  2"

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

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

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

$N_{21}$  jobs with processing order "2  $\rightarrow$  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  $\pi = (\pi^1, \pi^2)$  (two 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  $\rightarrow$  2"

$N_{21}$  jobs with processing order "2  $\rightarrow$  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  $\pi = (\pi^1, \pi^2)$  (two schedules, for machines 1 and 2 respectively)

Objective function: minimum total processing duration (makespan)

$$\min_{\pi} \max_{i \in \{1,2\}, j \in N} \{C_j^i(\pi)\}, \quad N = N_1 \cup N_2 \cup N_{12} \cup N_{21}$$

# 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.

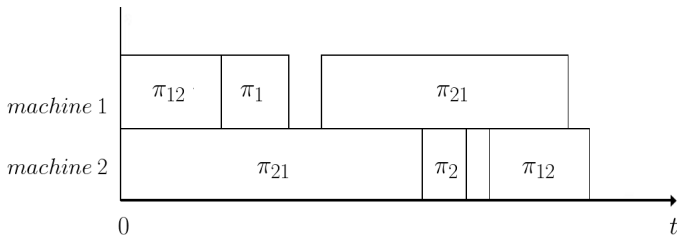
Without going into any deep detail on this problem, let us formulate the following theorem:

**Theorem.** Let  $\pi_{12}$  be a permutation of jobs obtained by applying Algorithm 1 to set of jobs  $N_{12}$ , let  $\pi_{21}$  be a permutation of jobs obtained by applying Algorithm 1 to set of jobs  $N_{21}$  (by swapping the machines), and let  $\pi_1$  and  $\pi_2$  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  $\rightarrow$  2  $\rightarrow$  3" "2  $\rightarrow$  3  $\rightarrow$  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  $\rightarrow$  2  $\rightarrow$  3" "2  $\rightarrow$  3  $\rightarrow$  1" and so on.

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

# Computational complexity of Jackson's, Smith's and Johnson'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 mentioned 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 mentioned before, Jackson's, Smith's and Johnson's problems have objective functions  $L_{max}$ ,  $\sum C_j$  and  $C_{max}$ , correspondingly.

**Try to answer these questions:** What is the purpose of these objective functions? Why are they exactly the way they are? Do they possibly reflect some traits of human behavior or are they just mathematical constructions?

## Problem of two production lines

# Problem of two production lines

1 job (product) that should pass  $n$  stages of production.

# Problem of two production lines

- 1 job (product) that should pass  $n$  stages of production.
- 2 production lines that both have  $n$  workplaces denoted as  $S_{11}, \dots, S_{1n}$  and  $S_{21}, \dots, S_{2n}$

# Problem of two production lines

1 job (product) that should pass  $n$  stages of production.

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

Each production stage  $j \in \{1, \dots, n\}$  is assigned one workplace  $S_{1j}$  at the 1st production line and one workplace  $S_{2j}$  at the 2nd production line. and it may be processed at any of them.

# Problem of two production lines

1 job (product) that should pass  $n$  stages of production.

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

Each production stage  $j \in \{1, \dots, n\}$  is assigned one workplace  $S_{1j}$  at the 1st production line and one workplace  $S_{2j}$  at the 2nd production line. and it may be processed at any of them.

After completion of stage  $j$ , the product is transferred either to workplace  $S_{1j+1}$  or  $S_{2j+1}$ , i. e. change of current production line is allowed.

# Problem of two production lines

1 job (product) that should pass  $n$  stages of production.

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

Each production stage  $j \in \{1, \dots, n\}$  is assigned one workplace  $S_{1j}$  at the 1st production line and one workplace  $S_{2j}$  at the 2nd production line. and it may be processed at any of them.

After completion of stage  $j$ , the product is transferred either to workplace  $S_{1j+1}$  or  $S_{2j+1}$ , i. e. change of current production line is allowed.

$a_{ij}$  — processing time at workplace  $S_{ij}$ ,  $i \in \{1, 2\}$ ,  $j \in \{1, \dots, n\}$

# Problem of two production lines

1 job (product) that should pass  $n$  stages of production.

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

Each production stage  $j \in \{1, \dots, n\}$  is assigned one workplace  $S_{1j}$  at the 1st production line and one workplace  $S_{2j}$  at the 2nd production line. and it may be processed at any of them.

After completion of stage  $j$ , the product is transferred either to workplace  $S_{1j+1}$  or  $S_{2j+1}$ , i. e. change of current production line is allowed.

$a_{ij}$  — processing time at workplace  $S_{ij}$ ,  $i \in \{1, 2\}$ ,  $j \in \{1, \dots, n\}$

$t_{ij}$  — transfer time from workplace  $S_{ij}$  to workplace  $S_{(3-i)j+1}$  (here  $(3-i)$  means that current production line is changed),

$i = 1, 2$ ,  $j \in \{1, \dots, n-1\}$ .

If current production line isn't changed, transfer time is 0.

Restrictions:

- 1 Processing of any production stage cannot start before previous stage is completed.

Restrictions:

- 1 Processing of any production stage cannot start before previous stage is completed.
- 2 Processing of any production stage cannot be interrupted.

# Problem of two production lines

Schedule  $\pi$  is a combination of  $n$  1's and 2's (each production stage is assigned to a workplace at the corresponding production line)

# Problem of two production lines

Schedule  $\pi$  is a combination of  $n$  1's and 2's (each production stage is assigned to a workplace at the corresponding production line)

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

Objective function: minimum total processing duration (makespan):

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

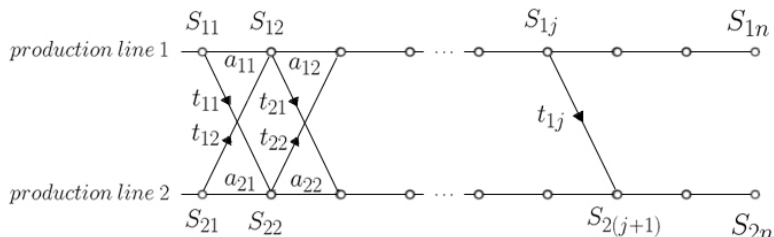
# Problem of two production lines

Schedule  $\pi$  is a combination of  $n$  1's and 2's (each production stage is assigned to a workplace at the corresponding production line)

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

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  $\pi$  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  $i$ .

$$C_1^1 := a_{11}$$

$$C_1^2 := a_{21}$$

for  $j := 2$  to  $n$  do

begin

$$C_j^1 := \min\{C_{j-1}^1, C_{j-1}^2 + t_{2(j-1)}\} + a_{1j}$$

$$C_j^2 := \min\{C_{j-1}^2, C_{j-1}^1 + t_{1(j-1)}\} + a_{2j}$$

end

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 what workplace the product came from to the current workplace.

Total computational complexity of this algorithm is  $O(n)$  operations.

# Meaning of objective functions

Earlier before, we have reviewed a number of problems with different types of objective functions (optimization criteria).

# Meaning of objective functions

Earlier before, we have reviewed a number of problems with different types of objective functions (optimization criteria).

The 2 main types of objective functions:

*min-max criteria* (e. g.  $\max_{j \in N} L_j(\pi) \rightarrow \min$ )

*summary criteria* (e. g.  $\sum_{j \in N} T_j(\pi) \rightarrow \min$ )

# Meaning of objective functions

**Min-max criteria** reflect the *"interests of the executor"*. For example, people tend to work first on task that has the closest due date. It may result in greater average delay, but the maximal delay is minimized.

Remember how students prepare for their finals? Let's imagine you have 3 exams and 3 days to prepare for each one. If you study the subject that has the closest deadline, you have exactly 3 days for each subject. Imagine that as a result your grades will be something like "B B B" (4, 4, 4). Now let's imagine that you study the first subject for 1 day, and the other two for 4 days. As a result your grades will be something like "C A A". But the latter technique is not acceptable if you don't want to lose your stipend.

# Meaning of objective functions

**Summary criteria** reflect the *"interests of the manager"*. For example, if each delayed task results in loss of profit, the manager aims to minimize total losses, thus minimizing the number of delayed tasks and the value of total penalty. It may result in greater maximal delay, but the average delay is minimized.

Imagine you're running a small car company that makes cars one at a time. Each client calls a day when he wants his car to be ready (due date). If your company is not able to produce a car in time, you drop the price to calm the client. To minimize loss of profit, you need to, obviously, minimize the total tardiness of all the orders. As a result, some cars may be finished in advance, and some may be delayed by a month or two, which in reality will probably result in some clients getting really angry and paying nothing at all. That's what the deadlines are for: the due dates can be violated, resulting in penalty, but the deadlines should never be violated at all.

# Scheduling Theory and Applications. Part I

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



V.A. TRAPEZNIKOV  
INSTITUTE  
OF CONTROL  
SCIENCES  
OF RUSSIAN ACADEMY  
OF SCIENCES



Thank you for your attention!