Scheduling

Radek Mařík

FEE CTU, K13132

April 28, 2015

Outline

1. Introduction to Scheduling
   - Methodology Overview
2. Classification of Scheduling Problems
   - Machine environment
   - Job Characteristics
   - Optimization
3. Local Search Methods
   - General
   - Tabu Search
   - Flow Shop Scheduling
4. Project Scheduling
   - Critical Path Method

Outline

1. Introduction to Scheduling
   - Methodology Overview
2. Classification of Scheduling Problems
   - Machine environment
   - Job Characteristics
   - Optimization
3. Local Search Methods
   - General
   - Tabu Search
   - Flow Shop Scheduling
4. Project Scheduling
   - Critical Path Method

Time, schedules, and resources [RN10]

- Classical planning representation
  - What to do
  - What order
- Extensions
  - How long an action takes
  - When it occurs
- Scheduling
  - Temporal constraints,
  - Resource contraints.
- Examples
  - Airline scheduling,
  - Which aircraft is assigned to which flights
  - Departure and arrival time,
  - A number of employees is limited.
  - An aircraft crew, that serves during one flight, cannot be assigned to another flight.

General Approach [Rud13]

Introduction

- Graham's classification of scheduling problems

General solving methods

- Exact solving method
  - Branch and bound methods
- Heuristics
  - dispatching rules
  - beam search
  - local search:
    simulated annealing, tabu search, genetic algorithms
- Mathematical programming: formulation
  - linear programming
  - integer programming
- Constraing satisfaction programming

Schedule [Rud13]

Schedule:

- determined by tasks assignments to given times slots using given resources, where the tasks should be performed

Complete schedule:

- all tasks of a given problem are covered by the schedule

Partial schedule:

- some tasks of a given problem are not resolved/assigned

Consistent schedule:

- a schedule in which all constraints are satisfied w.r.t. resource and tasks, e.g.
  - at most one tasks is performed on a signel machine with a unit capacity

Consistent complete schedule vs. consistent partial schedule

Optimal schedule:

- the assigments of tasks to machines is optimal w.r.t. to a given optimization criterion, e.g..
  - min C_{max}: makespan (completion time of the last task) is minimum

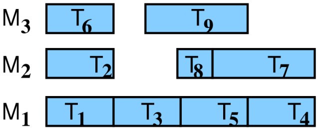
Terminology of Scheduling [Rud13]

Scheduling

concerns optimal allocation or assignment of resources, to a set of tasks or activities over time

- limited amount of resources,
- gain maximization given constraints

- Machines $M_i, i = 1, \dots, m$
- Jobs $J_j, j = 1, \dots, n$
- (i, j) **an operation** or processing of jobs j on machine i
  - a job can be composed from several operations,
  - example: job 4 has three operations with non-zero processing time (2,4),(3,4),(6,4), i.e. it is performed on machines 2,3,6



Static and dynamic parameters of jobs [Rud13]

- Static parameters of job
  - **processing time** p_{ij}, p_j:
    processing time of job j on machine i
  - **release date of** j r_j:
    earliest starting time of jobs j
  - **due date** d_j:
    committed completion time of job j (preference)
  - vs. **deadline**:
    time, when job j must be finished at latest (requirement)
  - **weight** w_j:
    importance of job j relatively to other jobs in the system
- Dynamic parameters of job
  - **start time** S_{ij}, S_j:
    time when job j is started on machine i
  - **completion time** C_{ij}, C_j:
    time when job j execution on machine i is finished

Static and dynamic parameters of jobs [Rud13]

- Static parameters of job
  - **processing time** p_{ij}, p_j:
    processing time of job j on machine i
  - **release date of** j r_j:
    earliest starting time of jobs j
  - **due date** d_j:
    committed completion time of job j (preference)
  - vs. **deadline**:
    time, when job j must be finished at latest (requirement)
  - **weight** w_j:
    importance of job j relatively to other jobs in the system
- Dynamic parameters of job
  - **start time** S_{ij}, S_j:
    time when job j is started on machine i
  - **completion time** C_{ij}, C_j:
    time when job j execution on machine i is finished

Outline

1. Introduction to Scheduling
   - Methodology Overview
2. Classification of Scheduling Problems
   - Machine environment
   - Job Characteristics
   - Optimization
3. Local Search Methods
   - General
   - Tabu Search
   - Flow Shop Scheduling
4. Project Scheduling
   - Critical Path Method

Graham's classification [Rud13, Nie10]

Graham's classification $\alpha|\beta|\gamma$

(Many) Scheduling problems can be described by a three field notation

- α: the machine environment
  - describes a way of job assingments to machines
- β: the job characteristics,
  - describes constraints applied to jobs
- γ: the objective criterion to be minimized
- complexity for combinations of scheduling problems

Examples

- $P3|prec|C_{max}$: bike assembly
- $Pm|r_j|\sum w_j C_j$: parallel machines

Graham's classification [Rud13, Nie10]

Graham's classification $\alpha|\beta|\gamma$

(Many) Scheduling problems can be described by a three field notation

- α: the machine environment
  - describes a way of job assingments to machines
- β: the job characteristics,
  - describes constraints applied to jobs
- γ: the objective criterion to be minimized
- complexity for combinations of scheduling problems

Examples

- $P3|prec|C_{max}$: bike assembly
- $Pm|r_j|\sum w_j C_j$: parallel machines

Machine Environment α [Rud13, Nie10]

- **Single machine ($\alpha = 1$):** $1|\ldots|\ldots$
- **Identical parallel machines** Pm
  - m identical machines working in parallel with the same speed
  - each job consist of a single operation,
  - each job processed by any of the machines m for p_j time units
- **Uniform parallel machines** Qm
  - processing time of job j on machine i propotional to its speed v_i
  - $p_{ij} = p_j / v_i$
  - ex. several computers with processor different speed
- **Unrelated parallel machines** Rm
  - machine have different speed for different jobs
  - machine i process job j with speed v_{ij}
  - $p_{ij} = p_j / v_{ij}$
  - ex. vector computer computes vector tasks faster than a classical PC

Machine Environment α [Rud13, Nie10]

- **Single machine ($\alpha = 1$):** $1|\ldots|\ldots$
- **Identical parallel machines** Pm
  - m identical machines working in parallel with the same speed
  - each job consist of a single operation,
  - each job processed by any of the machines m for p_j time units
- **Uniform parallel machines** Qm
  - processing time of job j on machine i propotional to its speed v_i
  - $p_{ij} = p_j/v_i$
  - ex. several computers with processor different speed
- **Unrelated parallel machines** Rm
  - machine have different speed for different jobs
  - machine i process job j with speed v_{ij}
  - $p_{ij} = p_j/v_{ij}$
  - ex. vector computer computes vector tasks faster than a classical PC

Machine Environment α [Rud13, Nie10]

- **Single machine ($\alpha = 1$):** $1|\ldots|\ldots$
- **Identical parallel machines** *Pm*
  - m identical machines working in parallel with the same speed
  - each job consist of a single operation,
  - each job processed by any of the machines m for p_j time units
- **Uniform parallel machines** *Qm*
  - processing time of job j on machine i propotional to its speed v_i
  - $p_{ij} = p_j/v_i$
  - ex. several computers with processor different speed
- **Unrelated parallel machines** *Rm*
  - machine have different speed for different jobs
  - machine i process job j with speed v_{ij}
  - $p_{ij} = p_j/v_{ij}$
  - ex. vector computer computes vector tasks faster than a classical PC

Machine Environment α [Rud13, Nie10]

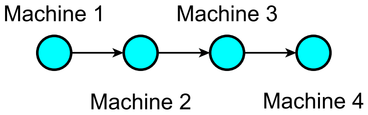
- **Single machine ($\alpha = 1$):** $1|\ldots|\ldots$
- **Identical parallel machines** *Pm*
  - m identical machines working in parallel with the same speed
  - each job consist of a single operation,
  - each job processed by any of the machines m for p_j time units
- **Uniform parallel machines** *Qm*
  - processing time of job j on machine i propotional to its speed v_i
  - $p_{ij} = p_j / v_i$
  - ex. several computers with processor different speed
- **Unrelated parallel machines** *Rm*
  - machine have different speed for different jobs
  - machine i process job j with speed v_{ij}
  - $p_{ij} = p_j / v_{ij}$
  - ex. vector computer computes vector tasks faster than a classical PC

Shop Problems [Rud13, Nie10]

- **Shop Problems**
  - each tasks is executed sequentially on several machine
    - job j consists of several operations (i, j)
    - operation (i, j) of jobs j is performed on machine i withing time p_{ij}
    - ex: job j with 4 operations $(1, j), (2, j), (3, j), (4, j)$
- Shop problems are classical
  studied in details in **operations research**
- Real problems are ofter more complicated
  - utilization of knowledge on subproblems or simplified problems in solutions

Shop Problems [Rud13, Nie10]

- **Shop Problems**
  - each tasks is executed sequentially on several machine
    - job j consists of several operations (i, j)
    - operation (i, j) of jobs j is performed on machine i withing time p_{ij}
    - ex: job j with 4 operations $(1, j), (2, j), (3, j), (4, j)$



- Shop problems are classical
  studied in details in **operations research**
- Real problems are ofter more complicated
  - utilization of knowledge on subproblems or simplified problems in solutions

Flow shop α [Rud13, Nie10]

- **Flow shop** *Fm*
  - m machines in series
  - each job has to be processed on each machine
  - all jobs follow the same route:
    - first machine 1, then machine 2, ...
  - if the jobs have to be processed in the same order on all machines, we have a **permutation** flow shop
- **Flexible flow shop** *FFs*
  - a generalizatin of flow shop problem
  - s phases, a set of parallel machines is assigned to each phase
  - i.e. flow shop with s parallel machines
  - each job has to be processed by all phase in the same order
    - first on a machine of phase 1, then on a machine of phase 2, ...
  - the task can be performed on any machine assigned to a given phase

Open shop & job shop [Rud13, Nie10]

- **Job shop** Jm
  - flow shop with m machines
  - each job has its individual predetermined route to follow
    - processing time of a given jobs might be zero for some machines
  - $(i, j) \rightarrow (k, j)$ specifies that job j is performed on machine i earlier than on machine k
  - example: $(2, j) \rightarrow (1, j) \rightarrow (3, j) \rightarrow (4, j)$
- **Open shop** Om
  - flow shop with m machines
  - processing time of a given jobs might be zero for some machines
  - no routing restrictions (this is a scheduling decision)

Outline

1. Introduction to Scheduling
   - Methodology Overview
2. Classification of Scheduling Problems
   - Machine environment
   - Job Characteristics
   - Optimization
3. Local Search Methods
   - General
   - Tabu Search
   - Flow Shop Scheduling
4. Project Scheduling
   - Critical Path Method

Constraints β [Rud13, Nie10]

- **Precedence constraints** *prec*
  - linear sequence, tree structure
  - for jobs a, b we write $a \to b$, with meaning of $S_a + p_a \leqslant S_b$
  - example: bike assembly
- **Preemptions** *pmtn*
  - a job with a higher priority interrupts the current job
- **Machine suitability** M_j
  - a subset of machines M_j, on which job j can be executed
  - room assignment: appropriate size of the classroom
  - games: a computer with a HW graphical library
- **Work force constraints** W, W_ℓ
  - another sort of machines is introduced to the problem
  - machines need to be served by operators and jobs can be performed only if operators are available, operators W
  - different groups of operators with a specific qualification can exist, W_ℓ is a number of operators in group ℓ

Constraints (continuation) β [Rud13, Nie10]

- **Routing constraints**
  - determine on which machine jobs can be executed,
  - an order of job execution in shop problems
    - job shop problem: an operation order is given in advance
    - open shop problem: a route for the job is specified during scheduling
- **Setup time and cost** $s_{ijk}, c_{ijk}, s_{jk}, c_{jk}$
  - depend on execution sequence
  - s_{ijk} time for execution of job k after job j on machine i
  - c_{ijk} cost of execution of job k after job j on machine i
  - s_{jk}, c_{jk} time/cost independent on machine
  - examples
    - lemonade filling into bottles
    - travelling salesman problem $1|s_{jk}|C_{max}$

Outline

1. Introduction to Scheduling
   - Methodology Overview
2. Classification of Scheduling Problems
   - Machine environment
   - Job Characteristics
   - Optimization
3. Local Search Methods
   - General
   - Tabu Search
   - Flow Shop Scheduling
4. Project Scheduling
   - Critical Path Method

Optimization: throughput and makespan γ [Rud13]

- **Makespan** C_{max}: maximum completion time

$$C_{max} = max(C_1, \ldots, C_n)$$

- Example: $C_{max} = max\{1, 3, 4, 5, 8, 7, 9\} = 9$
- Goal: **makespan minimization** often
  - maximizes **throughput**
  - ensures **uniform load of machines (***load balancing***)**
  - example: $C_{max} = max\{1, 2, 4, 5, 7, 4, 6\} = 7$

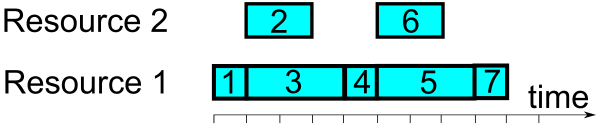
  - It is a basic criterion that is used very often.

Optimization: throughput and makespan γ [Rud13]

- **Makespan** C_{max}: maximum completion time

$$C_{max} = max(C_1, \ldots, C_n)$$

- Example: $C_{max} = max\{1, 3, 4, 5, 8, 7, 9\} = 9$



- Goal: **makespan minimization** often
  - maximizes **throughput**
  - ensures **uniform load of machines (*load balancing*)**
  - example: $C_{max} = max\{1, 2, 4, 5, 7, 4, 6\} = 7$

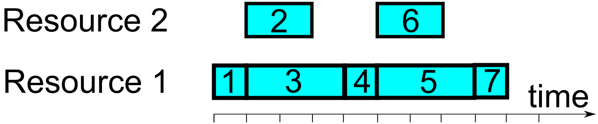
  - It is a basic criterion that is used very often.

Optimization: throughput and makespan γ [Rud13]

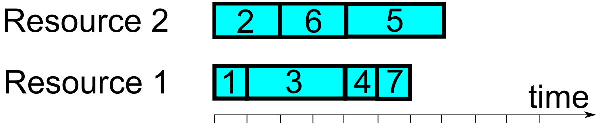
- **Makespan** C_{max}: maximum completion time

$$C_{max} = max(C_1, \ldots, C_n)$$

- Example: $C_{max} = max\{1, 3, 4, 5, 8, 7, 9\} = 9$



- Goal: **makespan minimization** often
  - maximizes **throughput**
  - ensures **uniform load of machines** (*load balancing*)
  - example: $C_{max} = max\{1, 2, 4, 5, 7, 4, 6\} = 7$



  - It is a basic criterion that is used very often.

Optimization: Lateness γ [Rud13]

- **Lateness** of job j: $L_{max} = C_j - d_j$
- **Maximum lateness** L_{max}

$$L_{max} = max(L_1, \ldots, L_n)$$

- Goal: **maximum lateness minimization**
- Example:

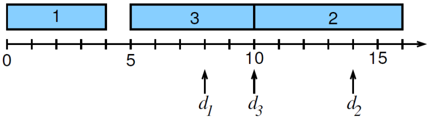
$$\begin{aligned} L_{max} &= max(L_1, L_2, L_3) = \\ &= max(C_1 - d_1, C_2 - d_2, C_3 - d_3) = \\ &= max(4 - 8, 16 - 14, 10 - 10) = \\ &= max(-4, 2, 0) = 2 \end{aligned}$$

Optimization: Lateness γ [Rud13]

- **Lateness** of job j: $L_{max} = C_j - d_j$
- **Maximum lateness** L_{max}

$$L_{max} = max(L_1, \ldots, L_n)$$

- Goal: **maximum lateness minimization**
- Example:



$$\begin{aligned} L_{max} &= max(L_1, L_2, L_3) = \\ &= max(C_1 - d_1, C_2 - d_2, C_3 - d_3) = \\ &= max(4 - 8, 16 - 14, 10 - 10) = \\ &= max(-4, 2, 0) = 2 \end{aligned}$$

Optimization: tardiness γ [Rud13]

- **Job tardiness** j: $T_j = max(C_j - d_j, 0)$
- **Total tardiness**

$$\sum_{j=1}^{n} T_j$$

- Goal: **total tardiness minimization**
- Example: $T_1 + T_2 + T_3 =$

$$\begin{aligned} &= \max(C_1 - d_1, 0) + \max(C_2 - d_2, 0) + \max(C_3 - d_3, 0) = \\ &= \max(4 - 8, 0) + \max(16 - 14, 0) + \max(10 - 10, 0) = \\ &= 0 + 2 + 0 = 2 \end{aligned}$$

- **Total weighted tardiness**

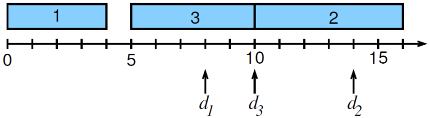
$$\sum_{j=1}^{n} w_j T_j$$

- Goal: **total weighted tardiness minimization**

Optimization: tardiness γ [Rud13]

- **Job tardiness** j: $\color{red}{T_j} = max(C_j - d_j, 0)$
- **Total tardiness**

$$\sum_{j=1}^{n} T_j$$



- Goal: **total tardiness minimization**
- Example: $T_1 + T_2 + T_3 =$

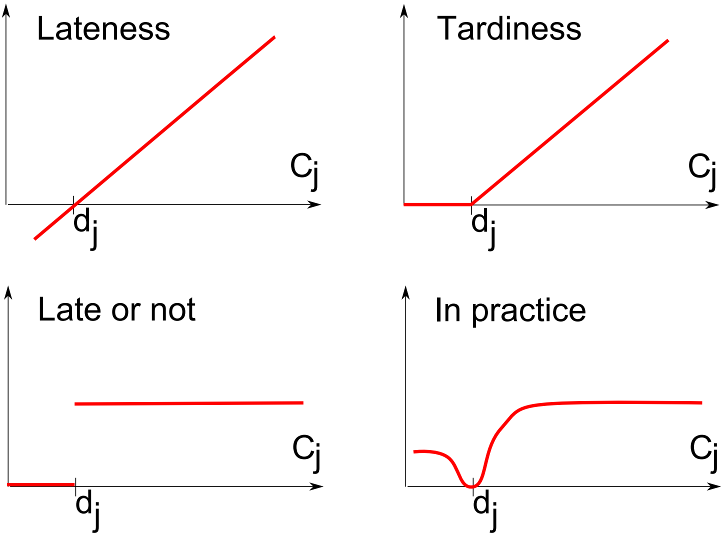
$$= \max(C_1 - d_1, 0) + \max(C_2 - d_2, 0) + \max(C_3 - d_3, 0) =$$
$$= \max(4 - 8, 0) + \max(16 - 14, 0) + \max(10 - 10, 0) =$$
$$= 0 + 2 + 0 = 2$$

- **Total weighted tardiness**

$$\sum_{j=1}^{n} w_j T_j$$

- Goal: **total weighted tardiness minimization**

Criteria Comparison γ [Rud13]



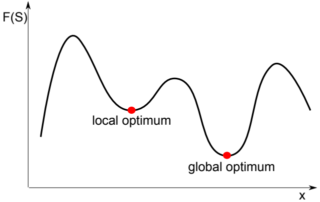
Outline

1. Introduction to Scheduling
   - Methodology Overview
2. Classification of Scheduling Problems
   - Machine environment
   - Job Characteristics
   - Optimization
3. Local Search Methods
   - General
   - Tabu Search
   - Flow Shop Scheduling
4. Project Scheduling
   - Critical Path Method

Constructive vs. local methods [Rud13]

- **Constructive methods**
  - Start with the empty schedule
  - Add step by step other jobs to the schedule so that the schedule remains consistent
- **Local search**
  - Start with a complete non-consistent schedule
    - trivial: random generated
  - Try to find a better "similar" schedule by local modifications.
  - Schedule quality is evaluated using optimization criteria
    - ex. makespan
  - optimization criteria assess also schedule consistency
    - ex. a number of vialoted precedence constraints
  - **Hybrid approaches**
    - combinations of both methods

Local Search Algorithm [Rud13]



1. **Initialization**
   - $k = 0$
   - Select an initial schedule S_0
   - Record the current best schedule: $S_{best} = S_0$ a $cost_{best} = F(S_0)$
2. **Select and update**
   - Select a schedule from neighborhood: $S_{k+1} \in N(S_k)$
   - if no element $N(S_k)$ satisfies schedule acceptance criterion then the algorithms finishes
   - if $F(S_{k+1}) < cost_{best}$ then $S_{best} = S_{k+1}$ a $cost_{best} = F(S_{k+1})$
3. **Finish**
   - if the stop constraints are satisfied then the algorithms finishes
   - otherwise $k = k + 1$ and continue with step 2.

Single machine + nonpreemptive jobs [Rud13]

- **Schedule representation**
  - permutations n jobs
  - example with six jobs: $1, 4, 2, 6, 3, 5$
- **Neighborhood definition**
  - pairwise exchange of neighboring jobs
    - $n-1$ possible schedules in the neighborhood
    - example: $1, 4, 2, 6, 3, 5$ is modified to $1, 4, 2, 6, 5, 3$
  - or select an arbitrary job from the schedule and place it to an arbitrary position
    - $\leqslant n(n-1)$ possible schedules in the neighborhood
    - example: from $1, 4, 2, 6, 3, 5$ we select randomly 4 and place it somewhere else: $1, 2, 6, 3, 4, 5$

Criteria for Schedule Selection [Rud13]

- Criteria for schedule selection
    - **Criterion for schedule acceptance/refuse**
- The main difference among a majority of methods
    - to accept a better schedule all the time?
    - to accept even worse schedule sometimes?
- methods
    - probabilistic
        - **random walk**: with a small probability (ex. 0.01) a worse schedule is accepted
        - **simulated annealing**
    - deterministic
        - **tabu search**: a tabu list of several last state/modifications that are not allowed for the following selection is maintained

Outline

1. Introduction to Scheduling
   - Methodology Overview
2. Classification of Scheduling Problems
   - Machine environment
   - Job Characteristics
   - Optimization
3. Local Search Methods
   - General
   - Tabu Search
   - Flow Shop Scheduling
4. Project Scheduling
   - Critical Path Method

Tabu Search [Rud13]

- **Deterministic criterion for schedule acceptance/refuse**
- **Tabu list** of several last schedule modifications is maintained
  - each new modification is stored on the top of the tabu list
    - ex. of a store modification: exchange of jobs j and k
  - tabu list = a list of forbidden modifications
  - the neighborhood is constrained over schedules, that do not require a change in the tabu list
    - a protection against cycling
    - example of a trivial cycling:
      the first step: exchange jobs 3 and 4, the second step: exchange jobs 4 and 3
  - a fixed length of the list (often: 5-9)
    - the oldest modifications of the tabu list are removed
    - too small length: cycling risk increases
    - too high length: search can be too constrained
- **Aspiration criterion**
  - determines when it is possible to make changes in the tabu list
  - ex. a change in the tabu list is allowed if $F(S_{best})$ is improved.

Tabu Search Algorithm [Rud13]

1. - $k = 1$
   - Select an initial schedule S_1 using a heuristics, $S_{best} = S_1$
2. - Choose $S_c \in N(S_k)$
   - If the modification $S_k \to S_c$ is forbidden because it is in the tabu list then continue with step 2
3. - If the modification $S_k \to S_c$ is not forbidden by the tabu list then $S_{k+1} = S_c$,
     store the reverse change to the top of the tabu list
     move other positions in the tabu list one position lower
     remove the last item of the tabu list
   - if $F(S_c) < F(S_{best})$ then $S_{best} = S_c$
4. - $k = k + 1$
   - if a stopping condition is satisfied then finish otherwise continue with step 2.

Example: tabu list [Rud13]

A schedule problem with $1|d_j|\sum w_j T_j$

- remind: $T_j = \max(C_j - d_j, 0)$

jobs	1	2	3	4
p_j	10	10	13	4
d_j	4	2	1	12
w_j	14	12	1	12

- Neighborhood: all schedules obtained by pair exchange of neighbor jobs
- Schedule selection from the neighborhood: select the best schedule
- Tabu list: pairs of jobs (j, k) that were exchanged in the last two modifications
- Apply tabu search for the initial solution $(2, 1, 4, 3)$
- Perform four iterations

Example: tabu list - solution I [Rud13]

jobs	1	2	3	4
p_j	10	10	13	4
d_j	4	2	1	12
w_j	14	12	1	12

$S_1 = (2, 1, 4, 3)$
$F(S_1) = \sum w_j T_j = 12 \cdot 8 + 14 \cdot 16 + 12 \cdot 12 + 1 \cdot 36 = 500 = F(S_{best})$
$F(1, 2, 4, 3) = 480$
$F(2, \underline{4}, \underline{1}, 3) = 436 = F(S_{best})$
$F(2, 1, 3, 4) = 652$
Tabu list: $\{(1, 4)\}$

$S_2 = (2, 4, 1, 3), F(S_2) = 436$
$F(\underline{4}, \underline{2}, 1, 3) = 460$
$F(2, 1, 4, 3)(= 500)$ tabu!
$F(2, 4, 3, 1) = 608$
Tabu list: $\{(2, 4), (1, 4)\}$

$S_3 = (4, 2, 1, 3), F(S_3) = 460$
$F(2, 4, 1, 3)(= 436)$ tabu!
$F(4, \underline{1}, \underline{2}, 3) = 440$
$F(4, 2, 3, 1) = 632$
Tabu list: $\{(2, 1), (2, 4)\}$

Example: tabu list - solution I [Rud13]

jobs	1	2	3	4
p_j	10	10	13	4
d_j	4	2	1	12
w_j	14	12	1	12

$S_1 = (2, 1, 4, 3)$
$F(S_1) = \sum w_j T_j = 12 \cdot 8 + 14 \cdot 16 + 12 \cdot 12 + 1 \cdot 36 = 500 = F(S_{best})$
$F(1, 2, 4, 3) = 480$
$F(2, \underline{4}, \underline{1}, 3) = 436 = F(S_{best})$
$F(2, 1, 3, 4) = 652$
Tabu list: $\{(1, 4)\}$

$S_2 = (2, 4, 1, 3), F(S_2) = 436$
$F(\underline{4}, \underline{2}, 1, 3) = 460$
$F(2, 1, 4, 3)(= 500)$ tabu!
$F(2, 4, 3, 1) = 608$
Tabu list: $\{(2, 4), (1, 4)\}$

$S_3 = (4, 2, 1, 3), F(S_3) = 460$
$F(2, 4, 1, 3)(= 436)$ tabu!
$F(4, \underline{1}, \underline{2}, 3) = 440$
$F(4, 2, 3, 1) = 632$
Tabu list: $\{(2, 1), (2, 4)\}$

Example: tabu list - solution I [Rud13]

jobs	1	2	3	4
p_j	10	10	13	4
d_j	4	2	1	12
w_j	14	12	1	12

$S_1 = (2,1,4,3)$
$F(S_1) = \sum w_j T_j = 12 \cdot 8 + 14 \cdot 16 + 12 \cdot 12 + 1 \cdot 36 = 500 = F(S_{best})$
$F(1,2,4,3) = 480$
$F(2,\underline{4},\underline{1},3) = 436 = F(S_{best})$
$F(2,1,3,4) = 652$
Tabu list: $\{(1,4)\}$

$S_2 = (2,4,1,3), F(S_2) = 436$
$F(\underline{4},\underline{2},1,3) = 460$
$F(2,1,4,3)(= 500)$ tabu!
$F(2,4,3,1) = 608$
Tabu list: $\{(2,4),(1,4)\}$

$S_3 = (4,2,1,3), F(S_3) = 460$
$F(2,4,1,3)(= 436)$ tabu!
$F(4,\underline{1},\underline{2},3) = 440$
$F(4,2,3,1) = 632$
Tabu list: $\{(2,1),(2,4)\}$

Example: tabu list - solution II [Rud13]

jobs	1	2	3	4
p_j	10	10	13	4
d_j	4	2	1	12
w_j	14	12	1	12

$S_3 = (4,2,1,3), F(S_3) = 460$
$F(2,4,1,3)(= 436)$ tabu!
$F(4,\underline{1},\underline{2},3) = 440$
$F(4,2,3,1) = 632$
Tabu list: $\{(2,1),(2,4)\}$

$S_4 = (4,1,2,3), F(S_4) = 440$
$F(\underline{1},\underline{4},2,3) = 408 = F(S_{best})$
$F(4,2,1,3)(= 460)$ tabu!
$F(4,1,3,2) = 586$
Tabu list: $\{(4,1),(2,1)\}$

$F(S_{best}) = 408$

Example: tabu list - solution II [Rud13]

jobs	1	2	3	4
p_j	10	10	13	4
d_j	4	2	1	12
w_j	14	12	1	12

$S_3 = (4, 2, 1, 3), F(S_3) = 460$
$F(2, 4, 1, 3)(= 436)$ tabu!
$F(4, \underline{1}, \underline{2}, 3) = 440$
$F(4, 2, 3, 1) = 632$
Tabu list: $\{(2, 1), (2, 4)\}$

$S_4 = (4, 1, 2, 3), F(S_4) = 440$
$F(\underline{1}, \underline{4}, 2, 3) = 408 = F(S_{best})$
$F(4, 2, 1, 3)(= 460)$ tabu!
$F(4, 1, 3, 2) = 586$
Tabu list: $\{(4, 1), (2, 1)\}$

$F(S_{best}) = 408$

Example: tabu list - solution II [Rud13]

jobs	1	2	3	4
p_j	10	10	13	4
d_j	4	2	1	12
w_j	14	12	1	12

$S_3 = (4, 2, 1, 3), F(S_3) = 460$
$F(2, 4, 1, 3)(= 436)$ tabu!
$F(4, \underline{1}, \underline{2}, 3) = 440$
$F(4, 2, 3, 1) = 632$
Tabu list: $\{(2, 1), (2, 4)\}$

$S_4 = (4, 1, 2, 3), F(S_4) = 440$
$F(\underline{1}, \underline{4}, 2, 3) = 408 = F(S_{best})$
$F(4, 2, 1, 3)(= 460)$ tabu!
$F(4, 1, 3, 2) = 586$
Tabu list: $\{(4, 1), (2, 1)\}$

$F(S_{best}) = 408$

Outline

1 Introduction to Scheduling
- Methodology Overview

2 Classification of Scheduling Problems
- Machine environment
- Job Characteristics
- Optimization

3 Local Search Methods
- General
- Tabu Search
- Flow Shop Scheduling

4 Project Scheduling
- Critical Path Method

Problem Statement [Pin09]

$$F2||C_{max}$$

Flow shop environment:

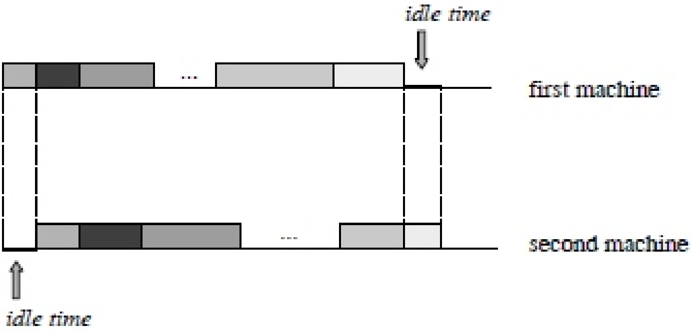
- 2 machines, n jobs
- objective function: makespan
- arrival times of jobs $r_j = 0$

- solution can be described by a sequence π
- problem was solved by Johnson in 1954

Johnson's Algorithm [Pin09]

1. Step 1. Schedule the group of jobs U that are shorter on the first machine than the second. $U = \ j \mid p1j < p2j$
2. Step 2. Schedule the group of jobs V that are shorter on the second machine than the first. $V = \ j \mid p1j \geqslant p2j$
3. Step 3. Arrange jobs in U in non-decreasing order by their processing times on the first machine.
4. Step 4. Arrange jobs in V in non-increasing order by their processing times on the second machine.
5. Step 5. Concatenate U and V and that is the processing order for both machines.

Johnson's Algorithm - sequence [Pin09]



Johnson's Algorithm - Example [Pin09]

Example.

jobs	1	2	3	4	5	6	7	8
p_{1j}	5	2	1	7	6	3	7	5
p_{2j}	2	6	2	5	6	7	2	1

$U = \{2, 3, 6\}$
$V = \{1, 4, 5, 7, 8\}$

jobs	3	2	6	5	4	7	1	8
p_{1j}	1	2	3	6	7	7	5	5
p_{2j}	2	6	7	6	5	2	2	1
C_{1j}	1	3	6	12	19	26	31	36
C_{2j}	3	9	16	22	27	29	33	37

$C_{max} = 37$

Outline

1 Introduction to Scheduling
- Methodology Overview

2 Classification of Scheduling Problems
- Machine environment
- Job Characteristics
- Optimization

3 Local Search Methods
- General
- Tabu Search
- Flow Shop Scheduling

4 Project Scheduling
- Critical Path Method

Problem Statement [Pin09]

- Environment:
  - parallel-machines,
  - jobs are subject to precedence constraints,
  - Objective: to minimize the makespan

$P\infty\|prec\|C_{max}$	$m \geqslant n$	Critical Path Method
$Pm\|prec\|C_{max}$	$2 \leqslant m < n$	NP hard

- **slack job**: the start of its processing time can be postponed without increasing the makespan,
- **critical job:** the job that can not be postponed,
- **critical path:** the set of critical jobs.

Critical Path Method [Pin09]

- **Forward procedure** that yields a schedule with minimum makespan.
- **Notation**
  - p_j ... processing time of jobs j
  - S'_j ... the earliest possible starting time of job j
  - C'_j ... the earliest possible completion time of job j
  - $C'_j = S'_j + p_j$
  - $\{\text{all } k \to j\}$ ... jobs that are predecessors of job j
- **Steps:**
  1. **Step 1** For each job j that has no predecessors $S'_j = 0$ and $C'_j = p_j$
  2. **Step 2** Compute inductively for each remaining job j

$$S'_j = \max_{\{\text{all } k \to j\}} C'_k$$

$$C'_j = S'_j + p_j$$

  3. **Step 3** $C_{max} = \max(C'_1, \ldots, C'_n)$

Critical Path Method II [Pin09]

- **Backward procedure** determines the latest possible starting and completion times.
- **Notation**
  - S''_j ...the latest possible starting time of job j
  - C''_j ...the latest possible completion time of job j
  - $\{j \rightarrow \text{all } k\}$ ...jobs that are successors of job j
- **Steps:**
  1. **Step 1**
     For each job j that has no successors $C''_j = C_{max}$ and $S''_j = C_{max} - p_j$
  2. **Step 2** Compute inductively for each remaining job j

$$C''_j = \min_{\{j \rightarrow \text{all } k\}} S''_k$$

$$S''_j = C''_j - p_j$$

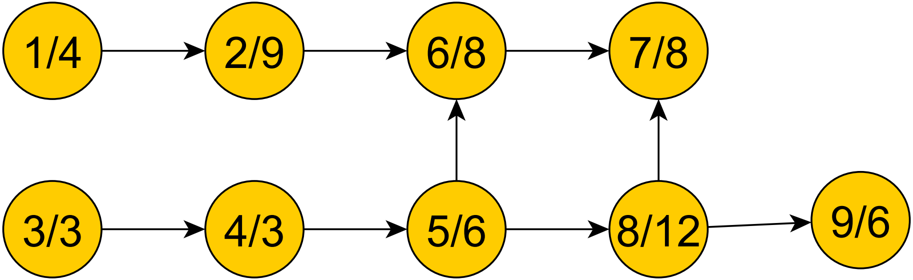
  3. **Step 3** Verify that $0 = \min(S''_1, \ldots, S''_n)$

Critical Path Method III [Pin09]

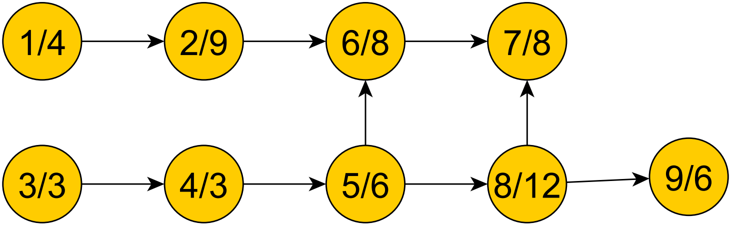
- The jobs whose earliest possible starting times are earlier than latest possible starting times are referred to as **slack jobs**.
- The jobs whose earliest possible starting times are equal to their latest possible starting times are **critical jobs**.
- A **critical path** is a chain of jobs which begin at time 0 and ends at C_{max}.

Critical Path Method - Example I [Pin09]

jobs	1	2	3	4	5	6	7	8	9
p_j	4	9	3	3	6	8	8	12	6

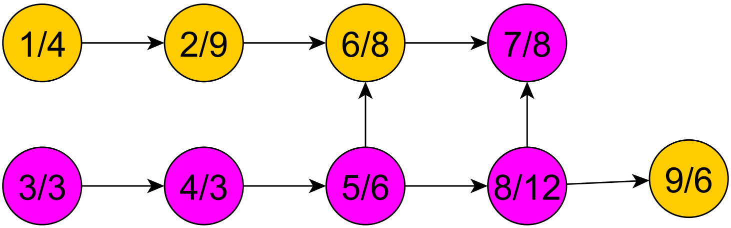


Critical Path Method - Example II [Pin09]



jobs	1	2	3	4	5	6	7	8	9
S_j'	0	4	0	3	6	max $\{13, 12\}$ =13	max $\{21, 24\}$ =24	12	24
C_j'	4	4+9 =13	3	3+3 =6	6+6 =12	13+8 =21	24+8 =32	12+12 =24	24+6 =30
C_j''	7	16	3	6	min $\{16, 12\}$ =12	24	32	min $\{24, 26\}$ =24	32
S_j''	7-4 =3	16-9 =7	3-3 =0	6-3 =3	12-6 =6	24-8 =16	32-8 =24	24-12 =12	32-6 =26

Critical Path Method - Example III [Pin09]



jobs	1	2	3	4	5	6	7	8	9
S'_j	0	4	0	3	6	max $\{13, 12\}$ $=13$	max $\{21, 24\}$ $=24$	12	24
C'_j	4	4+9 =13	3	3+3 =6	6+6 =12	13+8 =21	24+8 =32	12+12 =24	24+6 =30
C''_j	7	16	3	6	min $\{16, 12\}$ $=12$	24	32	min $\{24, 26\}$ $=24$	32
S''_j	7-4 =3	16-9 =7	3-3 =0	6-3 =3	12-6 =6	24-8 =16	32-8 =24	24-12 =12	32-6 =26

Critical Path Method - Extensions [Pin09]

- Stochastic activity (job) durations
- Nonavailability of resources
- Multiple resource types
- Preemption of activities
- Multiple projects with individual project due-dates

Objectives

- common one: minimising overall project duration
- *resource leveling* . . . minimise resource loading peaks without increasing project duration
- maximise resource utilisation factors

Critical Path Method - Extensions [Pin09]

- Stochastic activity (job) durations
- Nonavailability of resources
- Multiple resource types
- Preemption of activities
- Multiple projects with individual project due-dates

Objectives

- common one: minimising overall project duration
- *resource leveling* . . . minimise resource loading peaks without increasing project duration
- maximise resource utilisation factors

Literatura I

Tim Nieberg.
Lecture course "scheduling".
http://www.or.uni-bonn.de/lectures/ss10/sched10.html, July 2010.

Michael L. Pinedo.
Planning and Scheduling in Manufacturing and Services.
Springer-Verlag New York, 2 edition, 2009.

Stuart J. Russell and Peter Norvig.
Artificial Intelligence, A Modern Approach.
Pre, third edition, 2010.

Hana Rudová.
PA167 Rozvrhování, lecture notes, in Czech.
http://www.fi.muni.cz/ hanka/rozvrhovani/, March 2013.