# Scheduling

Radek Mařík

CVUT FEL, K13132

16. dubna 2014



## Content

## 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 fligths
  - 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

# Specific approach [Rud13]

- **Project planning:** project representation, critical path, time and cost trading, working force
- **Scheduling:** dispatching rules, branch and bound method, beam search,
- **Scheduling in manufacturing:** line with flexible time, with fixed time, with parallel working stations. m
- **Reservations:** interval scheduling, reservation system with reserves.
- **Timetabling:** scheduling with operators, scheduling with work force.
- **Scheduling of employees:** free day scheduling, work shift scheduling, cyclic shift scheduling.
- **University scheduling:** theory and practice





# 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

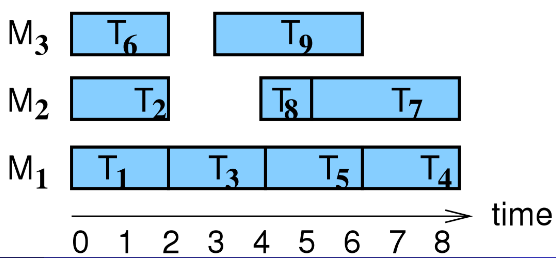
# Terminology of Scheduling [Rud13]

**Scheduling**

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

- limited amount of resources,
- gain maximization given constraints

- Machine $M_i, i = 1, \ldots, m$
- Jobs $J_j, j = 1, \ldots, n$
- $(i, j)$ **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





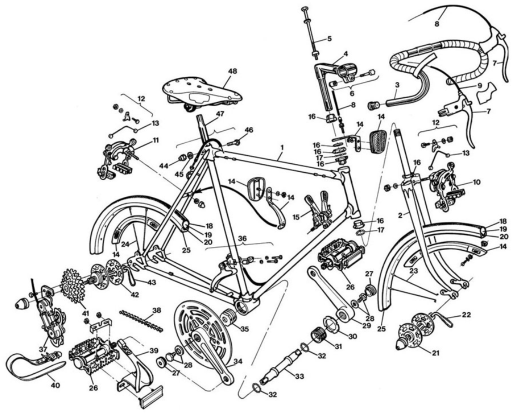

Machine oriented Gantt chart

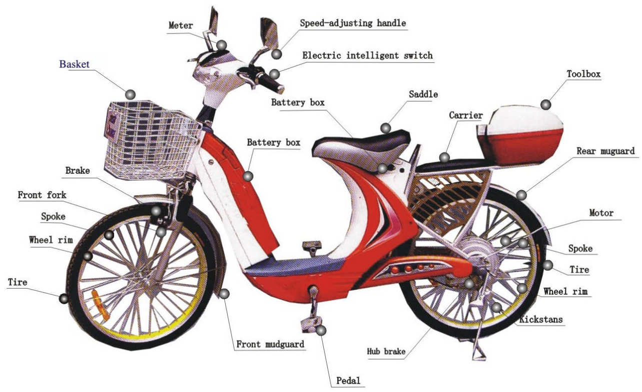


# Static and dynamic parameters of jobs [Rud13]

- 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 finised 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

# Example: bike assembly [Rud13]





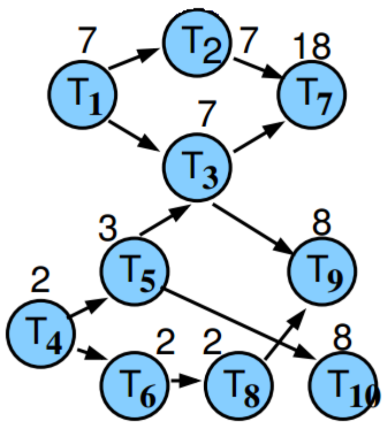

- 10 jobs with given processing time
- Precedence constraints
  - a given job can be executed after a specified subset of jobs
- Non-preemptive jobs
  - jobs cannot be interrupted
- Optimization criteria
  - makespan minimization
  - worker number minimization

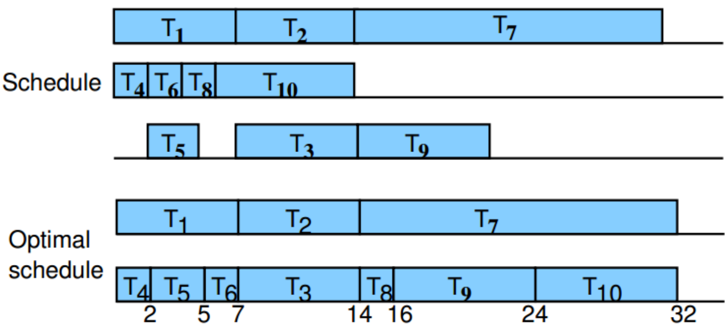




# Example: bike assembly [Rud13]

- 10 jobs with given processing time
- Precedence constraints
  - a given job can be executed after a specified subset of jobs
- Non-preemptive jobs
  - jobs cannot be interrupted
- Optimization criteria
  - makespan minimization
  - worker number minimization



## Scheduling Examples [Rud13]

- Scheduling of semiconductor manufacturing
  - a large amount of heterogenous products,
  - different amounts of produced items,
  - machine setup cost, required processing time guarantees
- Scheduling of supply chains
  - ex. a forest region – paper production – products from paper – distribution centers – end user
  - manufacturing cost, transport, storage minimization,
- Scheduling of paper production
  - input - wood, output - paper roles, expensive machines, different sorts of papers,
  - storage minimization
- Car assembly lines
  - manufacturing of different types of cars with different equipment,
  - throuput optimization, load balancing
- Lemonade filling into bottles
  - 4 flavors, each flavor has its own filling time,
  - cycle time minimization, one machine





## Scheduling Examples II [Rud13]

- Scheduling of hospital nurses
  - different numbers of nurses in working days and weekends,
  - weaker requirements for night shift rostering,
  - assignment of nurses to shifts, requirement satisfaction, cost minimization
- Grid computing scheduling
  - clusters, supercomputers, desktops, special devices,
  - scheduling of computation jobs and related resources,
  - scheduling of data transfers and data processing
- University scheduling
  - Time and rooms selection for subject education at universities
  - constraints given for subject placement,
  - preference requirements for time and room optimization,
  - minimization of overlapping subjects for all students,

# Scheduling vs. timetabling [Rud13]

## Scheduling . . . scheduling/planning

- resource allocation for given constraints over objects placed in time-space so that total cost of given resources is minimized,
- focus is given on object ordering, precedence conditions
  - ex. manufacturing scheduling: operation ordering determination, time dependencies of operation is important,
- **schedule**: specifies space and time information

## Timetabling

- resource allocation for given constraints over objects pakce in time-space so that given criteria are met as much as possible,
- focus is given on time placement of objects
- time horizon is often given in advance (a number of scheduled slots)
  - ex. education timetabling: time and a place is assigned to subjects
- **timetable**: shows when and where events are performed.



Introduction to Scheduling Terminology

# Sequencing and Rostering [Rud13]

## Sequencing

- for given constraints:
  - a construction of job order in which they will be executed
- **sequence**
  - an order in which jobs are executed
- ex. lemonade filling into bottles

## Rostering

- resource allocation for given constraints into slots using patterns
- **roster**
  - a list of person names, that determines what jobs are executed and when
- ex. a roster of hospital nurses, a roster of bus drivers

# Graham's classification [Rud13, Nie10]

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

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

- $\alpha$: the machine environment
  - describes a way of job assingments to machines
- $\beta$: the job characteristics,
  - describes constraints applied to jobs
- $\gamma$: 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



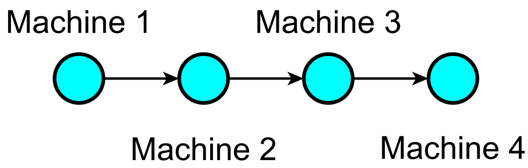


# Machine Environment $\alpha$ [Rud13, Nie10]

- **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





# Flow shop $\alpha$ [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 machine 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)





# Constraints $\beta$ [Rud13, Nie10]

- **Precedence constraints** *prec*
  - linear sequence, tree structure
  - for jobs $a, b$ we write $a \rightarrow b$, with meaning of $S_a + p_a \leq S_b$
  - 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) $\beta$ [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 $1|s_{jk}|C_{max}$



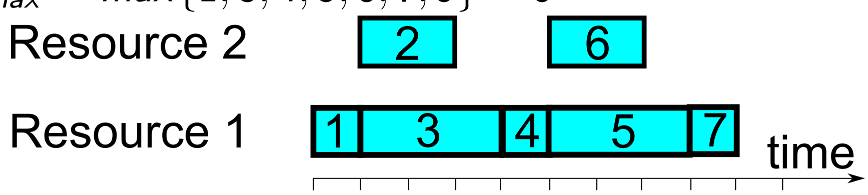


# Optimization: throughput and makespan $\gamma$ [Rud13]

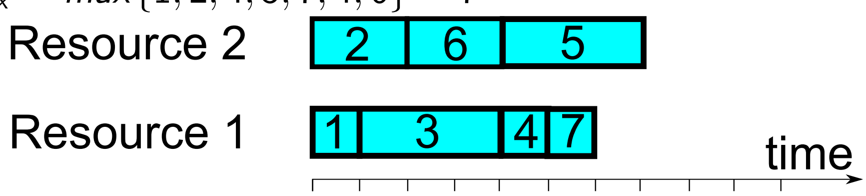
- **Makespan** $C_{max}$: maximum completion time

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

- 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: $C_{max} = max\{1, 2, 4, 5, 7, 4, 6\} = 7$



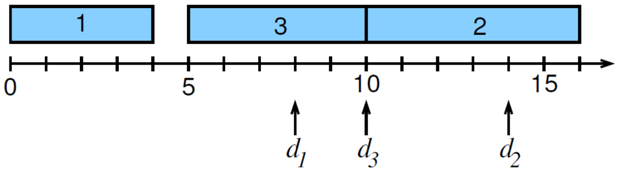

  - It is a basic criterion that is used very often.

# Optimization: Lateness $\gamma$ [Rud13]

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

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

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




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

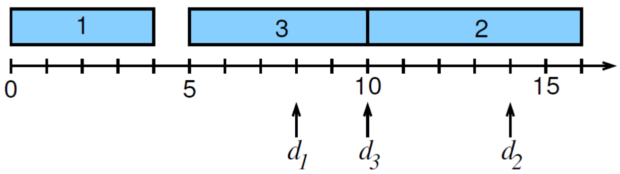




# Optimization: tardiness $\gamma$ [Rud13]

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

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




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

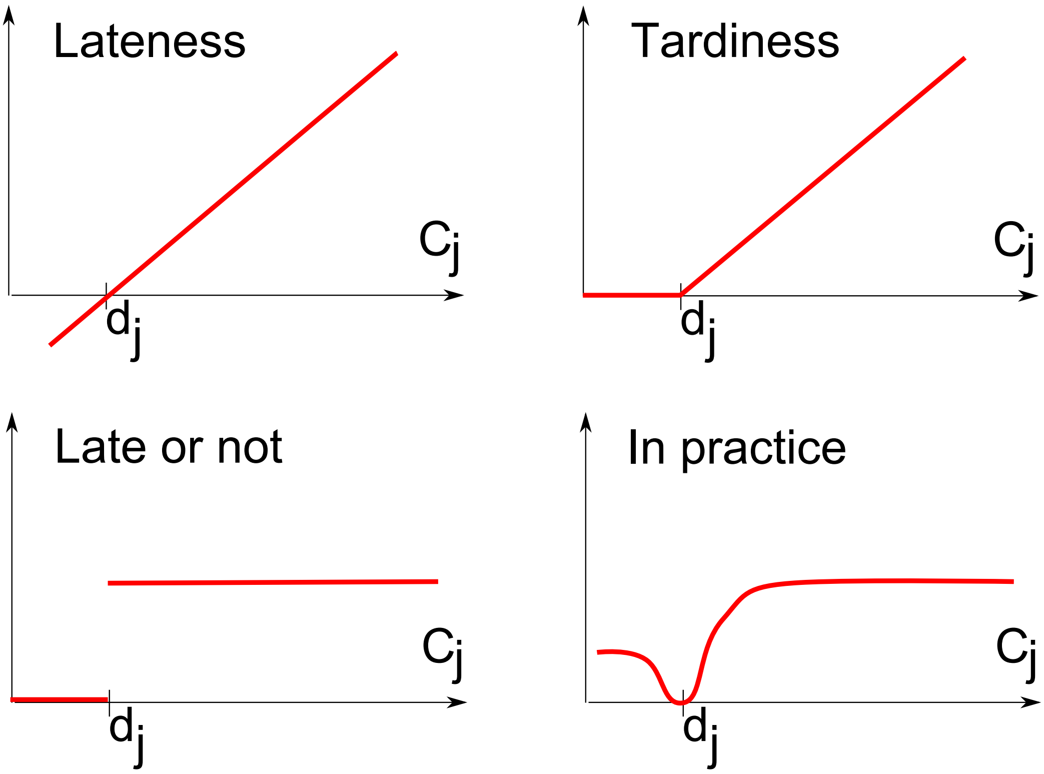
$$\begin{aligned} &= \max(C_1 - d_1, 0) + \max(C_2 - d_2, 0) + \max(C_3 - d_3, 0) = \\ &= \max(4 - 8, 0) + \max(16 - 14, 0) + \max(10 - 10, 0) = \\ &= 0 + 2 + 0 = 2 \end{aligned}$$

- **Total weighted tardiness**

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

- Goal: **total weighted tardiness minimization**

# Criteria Comparison $\gamma$ [Rud13]



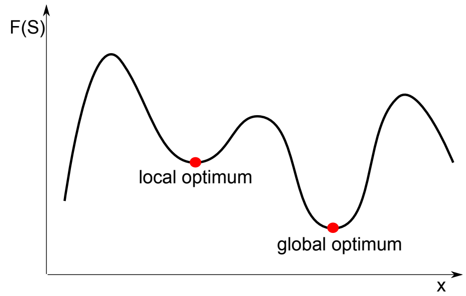





# Constructive vs. local methods [Rud13]

- **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
    - $\leq n(n - 1)$ 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





# 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 $S_k \rightarrow S_c$ is forbidden because it is in the tabu list then continue with step 2
3. 
   - If the modification $S_k \rightarrow S_c$ 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 schedule 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 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: $\{(4, 1), (2, 1)\}$

---

$F(S_{best}) = 408$

# Literatura I

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

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.