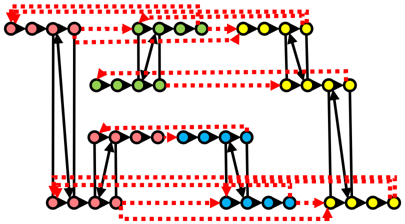
Modeling and solving job shop problems with complex process features and complicated objectives

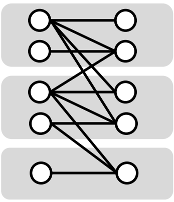
Reinhard Bürgy

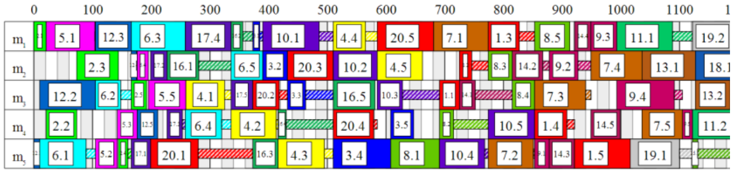
GERAD and École Polytechnique de Montréal

reinhard.burgy@gerad.ca

www. reinhardbuergy.ch







Seminar @ GERAD

Oct 27, 2016

Overview

1. The Classical Job Shop Scheduling Problem

1. Introduction
2. A Combinatorial Formulation in a Disjunctive Graph
3. Applications in Practice

2. Complex Process Features

1. Some Process Features
2. An Application: The BJS-RT and ALPHABOT

3. Complicated Objectives

1. General Regular Objective
2. A Class of Convex Cost Objectives

4. A Local Search Solution Approach

1. The Job Insertion Problem
2. Local Moves and Locally Improving Moves
3. Some Computational Results

Overview

1. The Classical Job Shop Scheduling Problem
   1. Introduction
   2. A Combinatorial Formulation in a Disjunctive Graph
   3. Applications in Practice
2. Complex Process Features
   1. Some Process Features
   2. An Application: The BJS-RT and ALPHABOT
3. Complicated Objectives
   1. General Regular Objective
   2. A Class of Convex Cost Objectives
4. A Local Search Solution Approach
   1. The Job Insertion Problem
   2. Local Moves and Locally Improving Moves
   3. Some Computational Results

1. The Classical Job Shop Scheduling Problem

1.1. Introduction

- The classical job shop scheduling problem
  - A fundamental optimization problem in operations research
  - NP-hard and one of the most computationally stubborn combinatorial problem (Applegate and Cook, 1991)
  - Addressed by numerous researchers, and a large body of knowledge accumulated over the last 60 years

- Given
  - A set M of machines (sometimes called resources)

    $M=\{m_1,m_2,m_3\}$
  - A set I of operations (sometimes called activities)

    $I=\{1,2,3,4,5,6,7,8,9,10\}$
  - A set $\mathcal{J}$ of jobs and

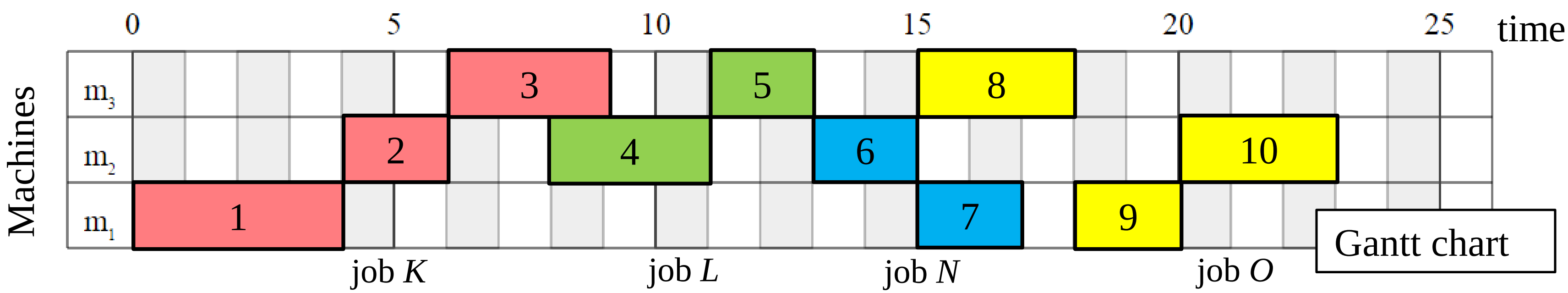
    $\mathcal{J}=\{K,L,N,O\}$
  - A job $J \in \mathcal{J}$ is an ordered set of operations $J = (J_1,J_2,\ldots J_{|J|})$, $J_i \in I$ for $i \in \{1,\ldots|J|\}$, specifying a processing sequence

    $K=(1,2,3)$ $L=(4,5)$ $N=(6,7)$ $O=(8,9,10)$
    - (Note: $\mathcal{J}$ is a partition of I, i.e. each operation is in exactly one job)
  - Each operation $i \in I$ executed on one machine m_i during p_i time units

- Data of the example:

Proc. time p	*First Op.*	*Sec. Op.*	*Third Op.*
Job K	m_1, 4	m_2, 2	m_3, 3
Job L	m_2, 3	m_3, 2	-
Job N	m_2, 2	m_1, 2	-
Job O	m_3, 3	m_1, 2	m_2, 3

- Find: A feasible schedule, i.e. a starting time of each operation:
  - Respect the processing sequences within the jobs
  - Each machine is used at most by one operation at any time ("unit capacity")
  - No preemption (no interruption) of the processing of an operation
- A visual representation of a schedule



- Objective: minimize the makespan (i.e., total duration)

A Problem Formulation

- Many different mathematical formulations exist
  - A main characteristic: continuous-time / discrete-time (time-indexed)
- Some references
  - Ku, Wen-Yang, and J. Christopher Beck. Mixed integer programming models for job shop scheduling: A computational analysis. *Computers & Operations Research* 73 (2016): 165-173
  - Brucker, P., & Knust, S. (2011). *Complex Scheduling*. Springer

"Introduction to Constraint Programming"
30 Oct - 1 Nov, Université Concordia

- A continuous-time formulation
  - Based on: Manne, A. (1960). On the job-shop scheduling problem. *Operations Research*, 8(2), 219–223
  - Introduce fictive end operation τ
  - Call two operation i and j *consecutive* if j follows i in some job
  - For each machine $m \in M$, let I_m be the set of operations executed on m
  - For each operation $i \in I \cup \{\tau\}$, α_i denotes the (variable) starting time

- A continuous-time formulation
  - Based on: Manne, A. (1960). On the job-shop scheduling problem. *Operations Research*, 8(2), 219–223
  - Introduce fictive end operation τ
  - Call two operation i and j *consecutive in a job* if j follows i in some job
  - For each machine $m \in M$, let I_m be the set of operations executed on m
  - For each operation $i \in I \cup \{\tau\}$, α_i denotes the (variable) starting time

Disjunctive programming formulation

$$
\begin{aligned}
&\text{minimize} \quad \alpha_\tau \\
&\text{subject to:} \\
&\alpha_j - \alpha_i \ge p_i \quad \text{for all } i \text{ and } j \text{ consecutive in a job} \\
&\alpha_\tau - \alpha_i \ge p_i \quad \text{for all } i \in I \\
&\alpha_j - \alpha_i \ge p_i \ \text{ OR } \ \alpha_i - \alpha_j \ge p_j \quad \text{for all } i,j \in I_m \\
&\alpha_i \ge 0 \quad \text{for all } i \in I \cup \{\tau\}
\end{aligned}
$$

min. end time

job structure

end operation

machine capacity

start after 0

replace OR operator by:

Mixed-integer linear progr. formulation

$$
\begin{aligned}
&\text{minimize} \quad \alpha_\tau \\
&\text{subject to:} \\
&\alpha_j - \alpha_i \ge p_i \quad \text{for all } i \text{ and } j \text{ consecutive in a job} \\
&\alpha_\tau - \alpha_i \ge p_i \quad \text{for all } i \in I \\
&\boxed{\begin{aligned}&\alpha_j - \alpha_i \ge p_i - M(1 - z_{ij}), \\ &\alpha_i - \alpha_j \ge p_j - M z_{ij} \quad \text{for all } i,j \in I_m\end{aligned}} \\
&\alpha_i \ge 0 \quad \text{for all } i \in i \in I \cup \{\tau\} \\
&\boxed{z_{ij} \in \{0,1\} \quad \text{for all } i,j \in I_m}
\end{aligned}
$$

z_{ij}: 1 if i is before j, and 0 otherwise
M: large constant, here e.g., $M = \sum_{i \in I} p_i$

1.2. A Combinatorial Formulation in a Disjunctive Graph

- Constraints have all the same structure: $\alpha_w - \alpha_v \geq d_{vw}$ Difference of two times

  Precedence Constraints

- Simplify the formulation by introducing a disjunctive graph

Disjunctive programming formulation

minimize $\alpha_\tau - \alpha_\sigma$

subject to: a constant

$\alpha_j - \alpha_i \geq p_i$ for all i and j consecutive in a job — conjunctive

$\alpha_\tau - \alpha_i \geq p_i$ for all $i \in I$

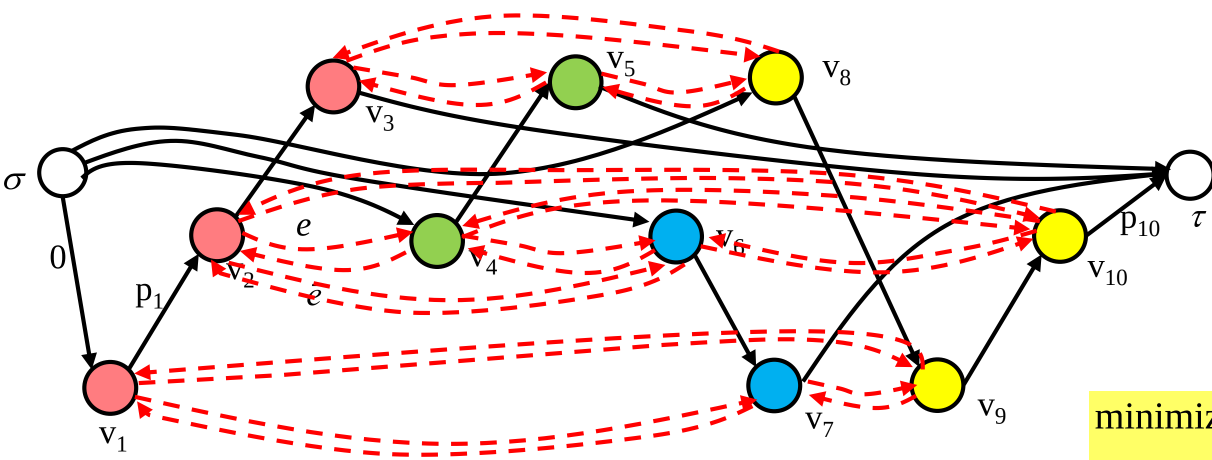
$\alpha_j - \alpha_i \geq p_i$ OR $\alpha_i - \alpha_j \geq p_j$ for all $i,j \in I_m$ — disjunctive

$\alpha_i \geq 0$ for all $i \in I \cup \{\tau\}$ — conjunctive

$-\alpha_\sigma$

(and $\alpha_\sigma = 0$, which is, actually, simple to fulfill)

$G=(V,A,E,\mathcal{E},d)$



(some redundant arcs incident to σ and τ omitted)

- Each operation is repres.by a node
- Each precedence constraint $x_w - x_v \geq d_{vw}$ represented by an arc (v,w) with weight d_{vw}
- Set of conjunctive arc A and
- Set of disjunctive arcs E
- Disjunctive structure $\mathcal{E}$: consists of pairs of disjunctive arcs $\{e, \bar{e}\} \in \mathcal{E}$

minimize $\alpha_\tau - \alpha_\sigma$

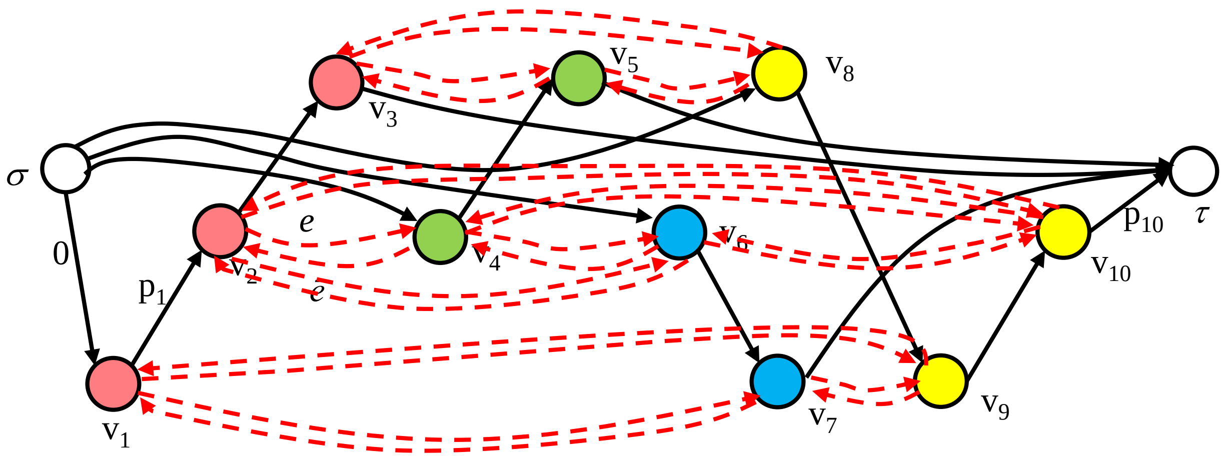
subject to:

$\alpha_w - \alpha_v \geq d_{vw}$ for all $(v,w) \in A$

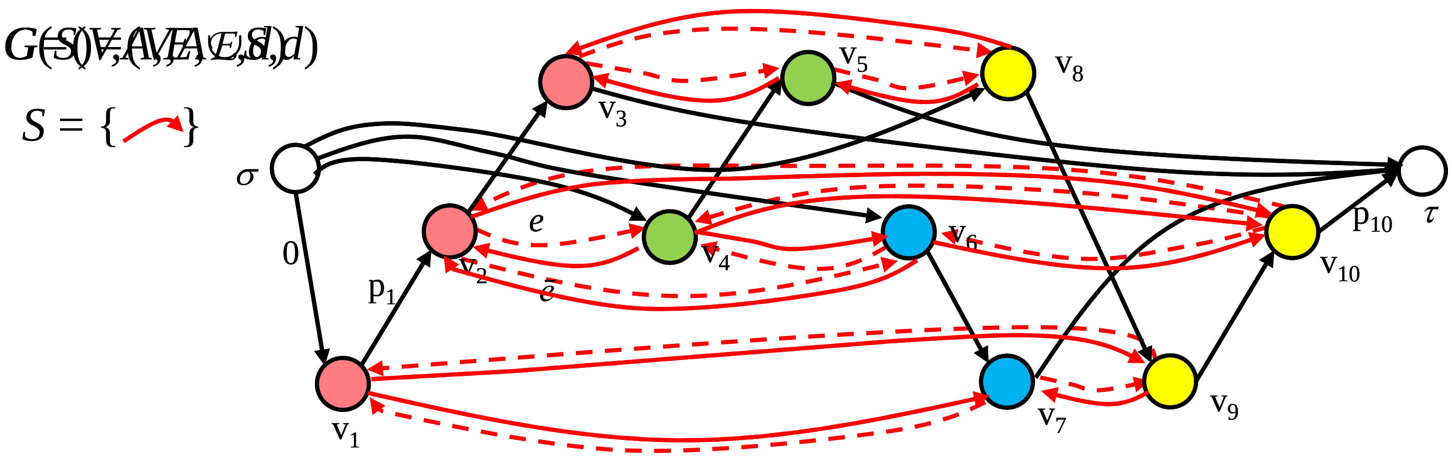
$\alpha_w - \alpha_v \geq d_{vw}$ OR $\alpha_{v'} - \alpha_{w'} \geq d_{v'w'}$

for all $\{(v,w),(v',w')\} \in \mathcal{E}$

$G=(V,A,E,\mathcal{E},d)$



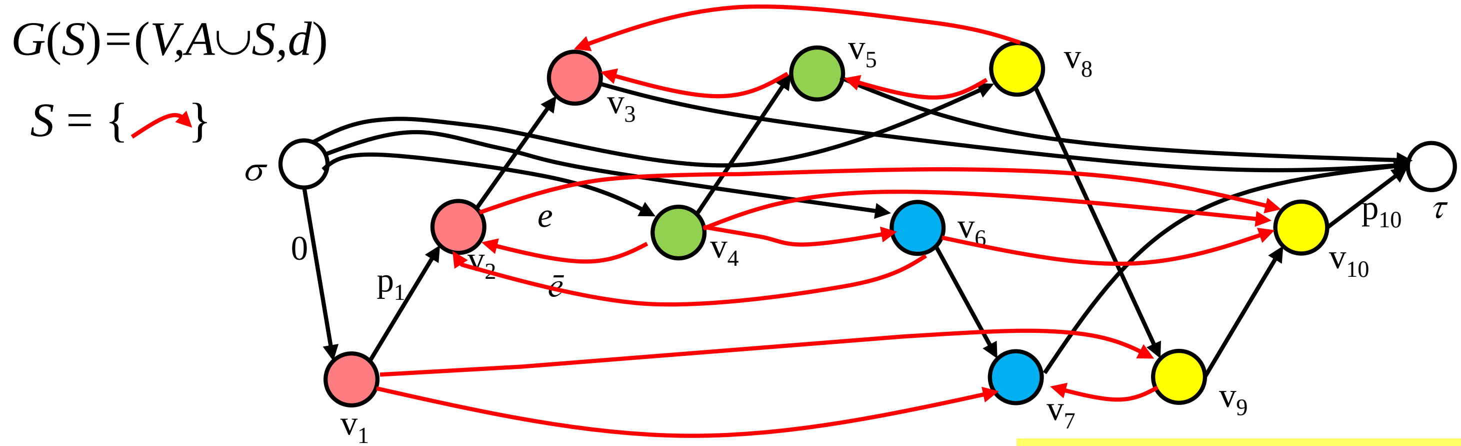
Selections



- Capturing solutions:

Definition
A selection: any set $S \subseteq E$ of disjunctive arcs
A selection S is complete if $S \cap \{e,\bar{e}\} \neq \emptyset$ for all $\{e,\bar{e}\} \in \mathcal{E}$
A selection S is positive acyclic if the graph G(S)=(V,$A\cup S$,d) has no cycle of positive length and positive cyclic otherwise.

Timing Problem: Determine Starting Times



- For any selection S, we have the following timing problem at hand:
  - A potential problem!

$$\text{minimize} \quad \alpha_\tau - \alpha_\sigma$$
$$\text{subject to:}$$
$$\alpha_w - \alpha_v \geq d_{vw} \quad \text{for all } (v,w) \in A \cup S$$

- This is a (special) LP with the following dual:

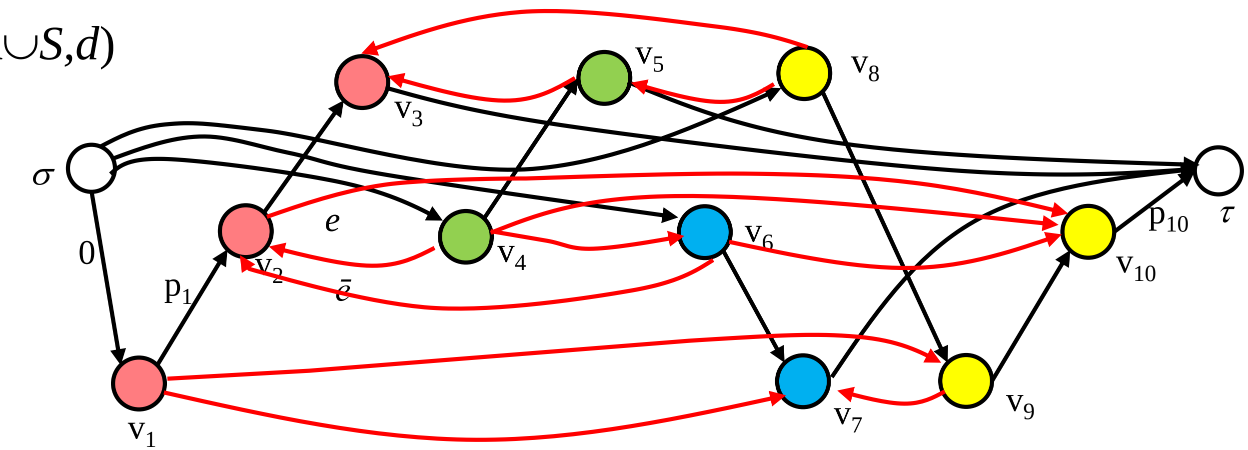
$$\text{maximize} \quad \sum_{(v,w)\in A\cup S} d_{vw} x_{vw}$$
$$\text{subject to:}$$
$$\sum_{w:(w,v)\in A\cup S} x_{wv} - \sum_{w:(v,w)\in A\cup S} x_{vw} = \begin{cases} 1 & \text{for } v = \tau \\ -1 & \text{for } v = \sigma \\ 0 & \text{for } v \in V - \{\sigma,\tau\} \end{cases}$$
$$x_{vw} \geq 0 \text{ for all } (v,w) \in A \cup S$$

A (simple) network flow problem!

- Well-known and easy to see: feasible times exist if and only if S is positive acyclic
- Hence, selection S called feasible if S is complete and positive acyclic

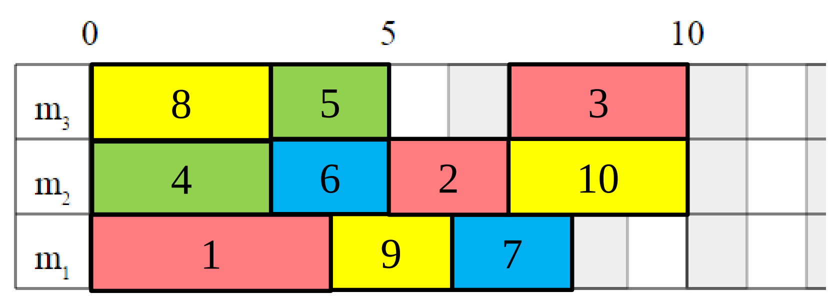
$G(S)=(V,A \cup S,d)$

$S = \{\ \}$



- For any feasible selection S, an optimal solution can be found by
  - Computing the earliest time schedule $\alpha(S)$ where
  - For all $v \in V$, $\alpha_v(S)$: length of a longest path from σ to node v in $G(S)$
  - Can be done by topological sorting algorithm, time complexity: $O(n+m)$
- Timing problem efficiently solvable!

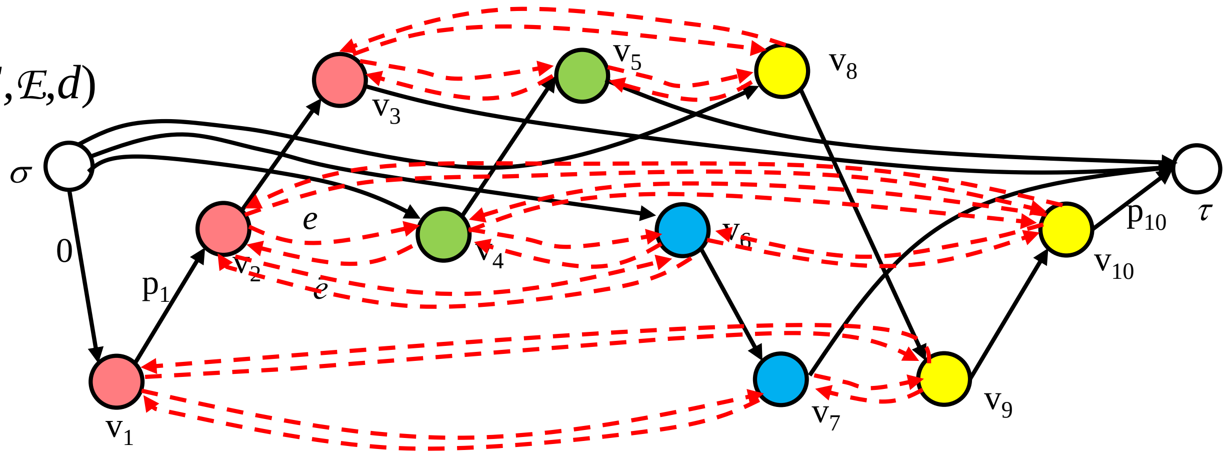
Earliest time schedule of the selection above:



Makespan: 10

A Combinatorial Problem Formulation

$G=(V,A,E,\mathcal{E},d)$



Among all feasible selections, find a selection S minimizing the length of a longest path from σ to τ in $G(S)$.

1.3. Applications in Practice





Liebherr: construction machines, factory in Telfs, Austria

Source: https://www.youtube.com/watch?v=V-37jTfLe8o

Flexible Manufacturing Systems and Robotic Cells

Flexible manufacturing system (Kuka)

Source: http://www.kuka-systems.com/NR/exeres/73B56636-DC3A-4CD8-A7E7-9D9E15345AB1

A robotic cell:

Source: http://www.canadianmetalworking.com/features/where-are-your-robots/

Logendran, R., & Sonthinen, A. (1997). A tabu search-based approach for scheduling job-shop type flexible manufacturing systems. *Journal of the Operational Research Society*, *48*(3), 264–277.

Hall, N. G., Kamoun, H., & Sriskandarajah, C. (1997). Scheduling in robotic cells: classification, two and three machine cells. *Operations Research*, *45*(3), 421–439.

Hoist Scheduling and Factory Cranes

Electroplating plant for surface treatment
Source: Surface Technology Solutions (stsindustrie.com)

A paper-roll shipping store with automated transports
Source: liftandhoist.com

Leung, J. M. Y., Zhang, G., Yang, X., Mak, R., & Lam, K. (2004). Optimal cyclic multi-hoist scheduling: a mixed integer programming approach. *Operations Research*, *52*(6), 965–976.

Peterson, B., Harjunkoski, I., Hoda, S., & Hooker, J. N. (2014). Scheduling multiple factory cranes on a common track. *Computers and Operations Research*, *48*, 102–112.

Gantry Crane Scheduling

A rail-road terminal with gantry cranes
Source: dradio.de

Automated stacking cranes in a storage area
Source: wcms.demagrobots.info

Li, W., Wu, Y., Petering, M. E. H., Goh, M., & de Souza, R. (2009). Discrete time model and algorithms for container yard crane scheduling. *European Journal of Operational Research*, *198*(1), 165–172.

Ng, W. C. (2005). Crane scheduling in container yards with inter-crane interference. *European Journal of Operational Research*, *164*(1), 64–78.

Train Scheduling

Trains
Source: www.thechronicle.com.au

Metro Stations
Source: en.wikipedia.org/wiki/Duomo_(Milan_Metro)

Liu, S., & Kozan, E. (2011). Scheduling trains with priorities: a no-wait blocking parallel-machine job-shop scheduling model. *Transportation Science*, *45*(2), 175–198

Mannino, C., & Mascis, A. (2009). Optimal Real-Time Traffic Control in Metro Stations. *Operations Research*, *57*(4), 1026–1039

Gap Between Theory and Practice

- However, the classical job shop rarely applicable in practice
- Process features that are not captured
  - Sequence-dependent setup times
    - Cleaning
    - Idle moving of robots
  - Routing flexibility
    - Multiple machines of the same type
  - Storage time restrictions
    - Chemical treatments
  - Storage space restrictions
    - Train is always on a rail section (= machines)
    - Robotic cells (predefined or no storage space)

- More complicated objectives than the makespan
  - Work-in process related
    - Total (weighted) flow time
  - Tardiness related
    - Total (weighted) linear tardiness costs
    - Total (weighted) squared tardiness costs
    - Number of tardy jobs
  - Earliness and tardiness related
    - "just-in-time objectives": sum of earliness and tardiness costs, possibly non-linear
- Numerous new job shop scheduling models appeared
  - Mostly, capturing just few features
  - Specific, application-oriented solution approaches
  - Job shop scheduling research has become fragmented, and scheduling software highly specialized (Bulbul, K., & Kaminsky, P. (2013). A linear programming-based method for job shop scheduling. Journal of Scheduling, 16(2), 161–183)
- Our approach: develop generic models and generic methods !

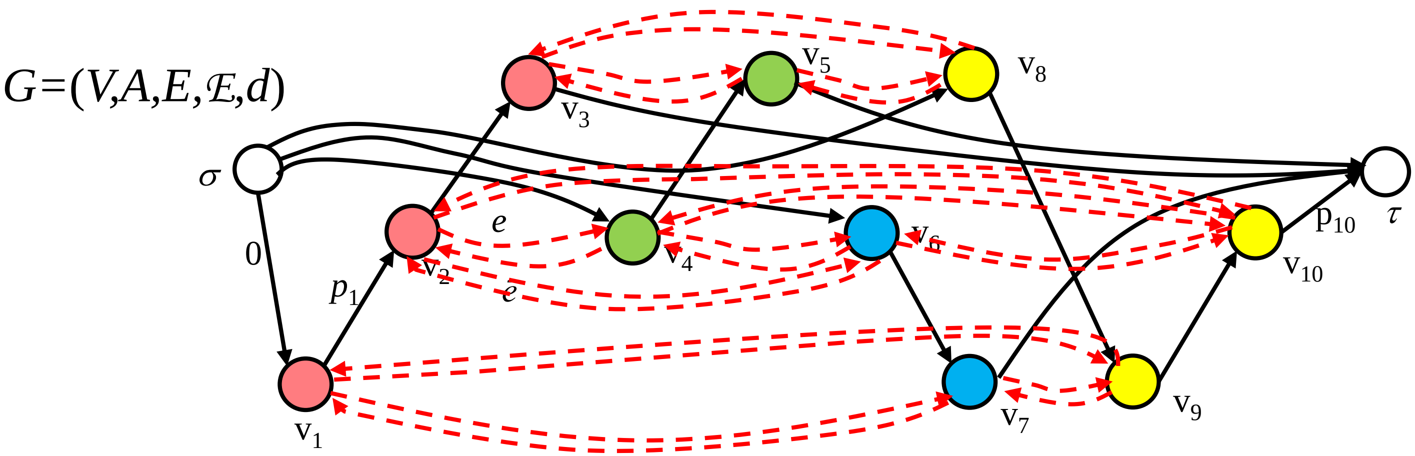
Goal of this talk: show what we do.

Overview

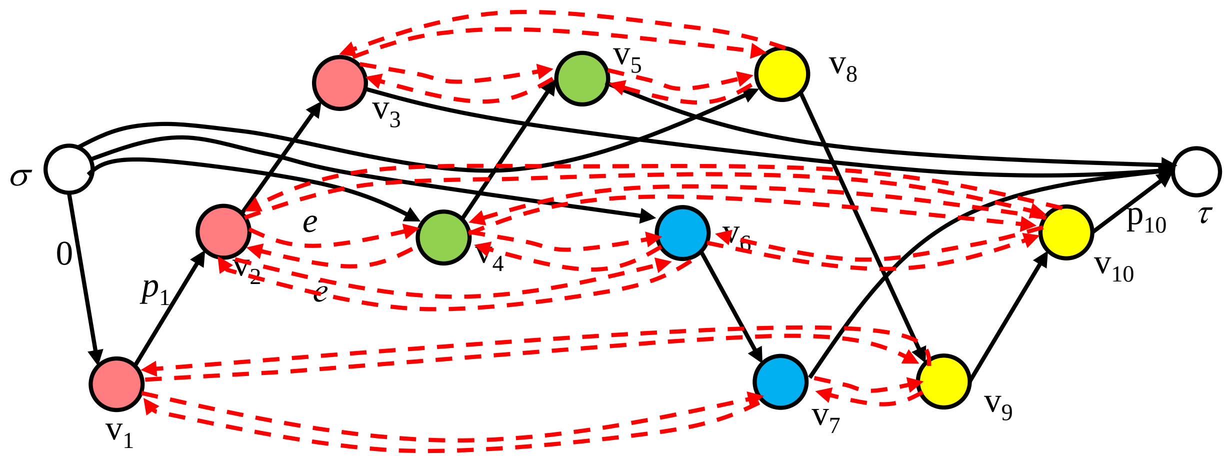
1. The Classical Job Shop Scheduling Problem
   1. Introduction
   2. A Combinatorial Formulation in a Disjunctive Graph
   3. Applications in Practice
2. Complex Process Features
   1. Simple Process Features
   2. More Complex Process Features
   3. An Application: The BJS-RT and ALPHABOT
3. Complicated Objectives
   1. General Regular Objective
   2. A Class of Convex Cost Objectives
4. A Local Search Solution Approach
   1. The Job Insertion Problem
   2. Local Moves and Locally Improving Moves
   3. Some Computational Results

2. Complex Process Features

- Incorporate additional process features in our disjunctive graph formulation



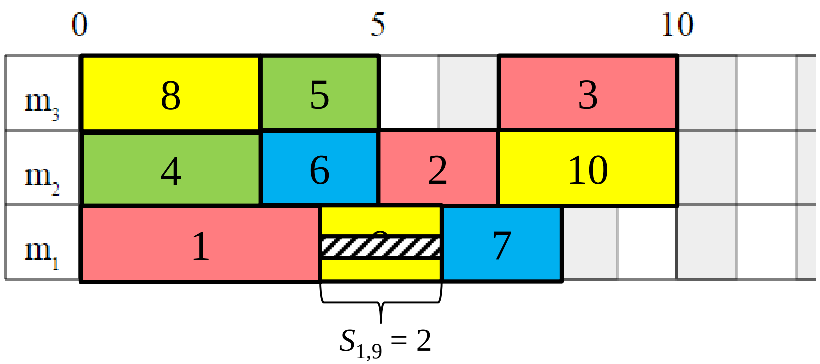
Among all feasible selections, find a selection S minimizing the length of a longest path from σ to τ in $G(S)$.

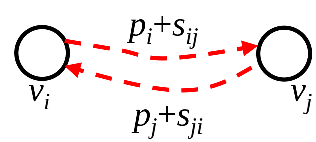


v5
v8
v3
σ
τ
e
v4
v6
p10
0
v2
v10
p1
e
v1
v7
v9

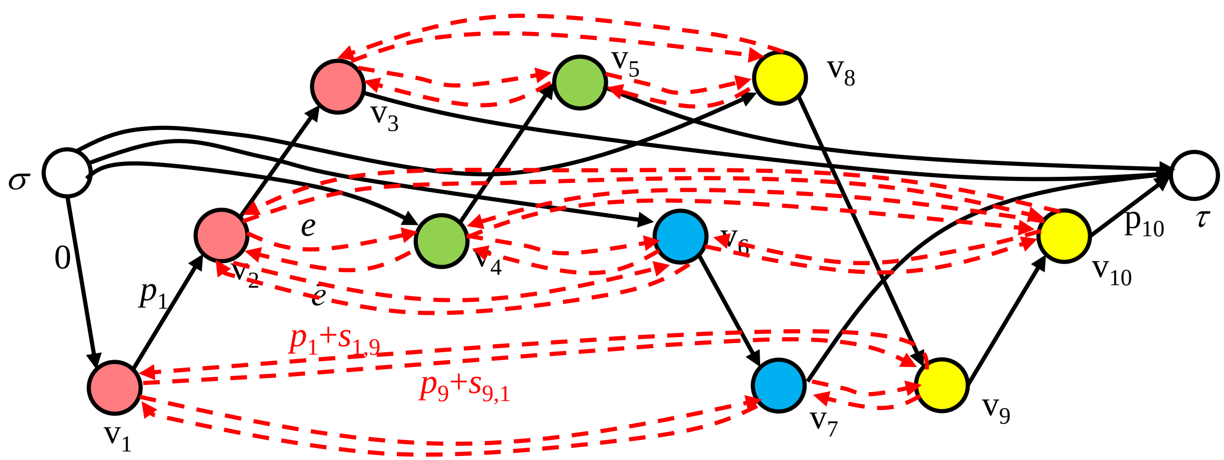
2.1. Some Process Features
Sequence-Dependent Setup Times

- If op. j directly follows op. i on some machine, then a setup of duration s_{ij} occurs between the completion of i and the start of j



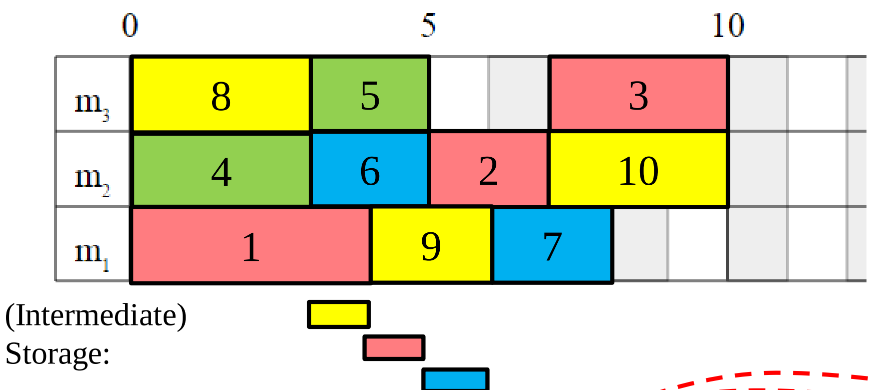


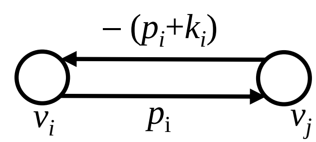
(Note: setups must satisfy so-called weak triangle inequalities.)



Limited Storage Time

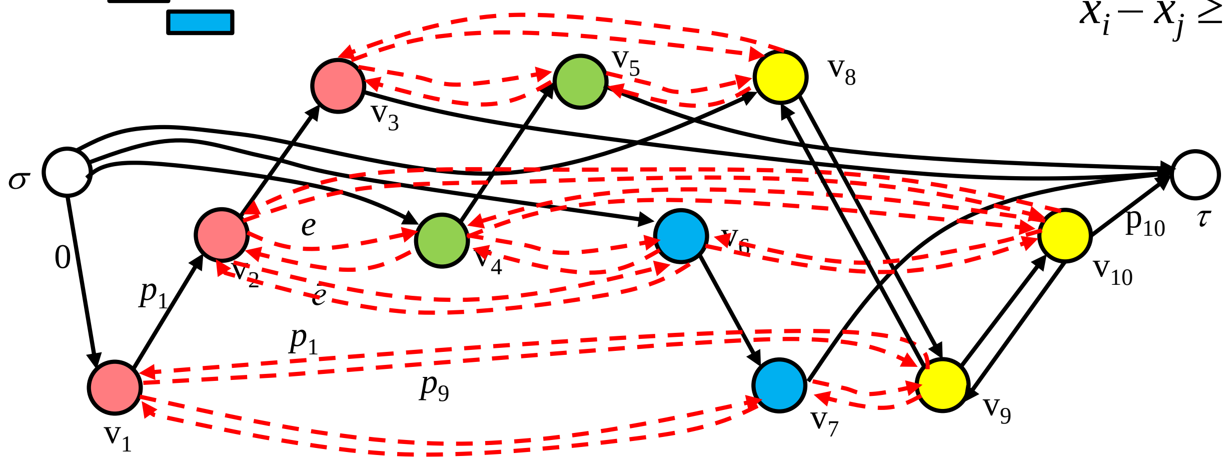
- After the completion of operation *i*, the job of *i* can be stored (or can wait) at most k_i time units before the processing of its next operation *j* starts (also called maximum time lag)





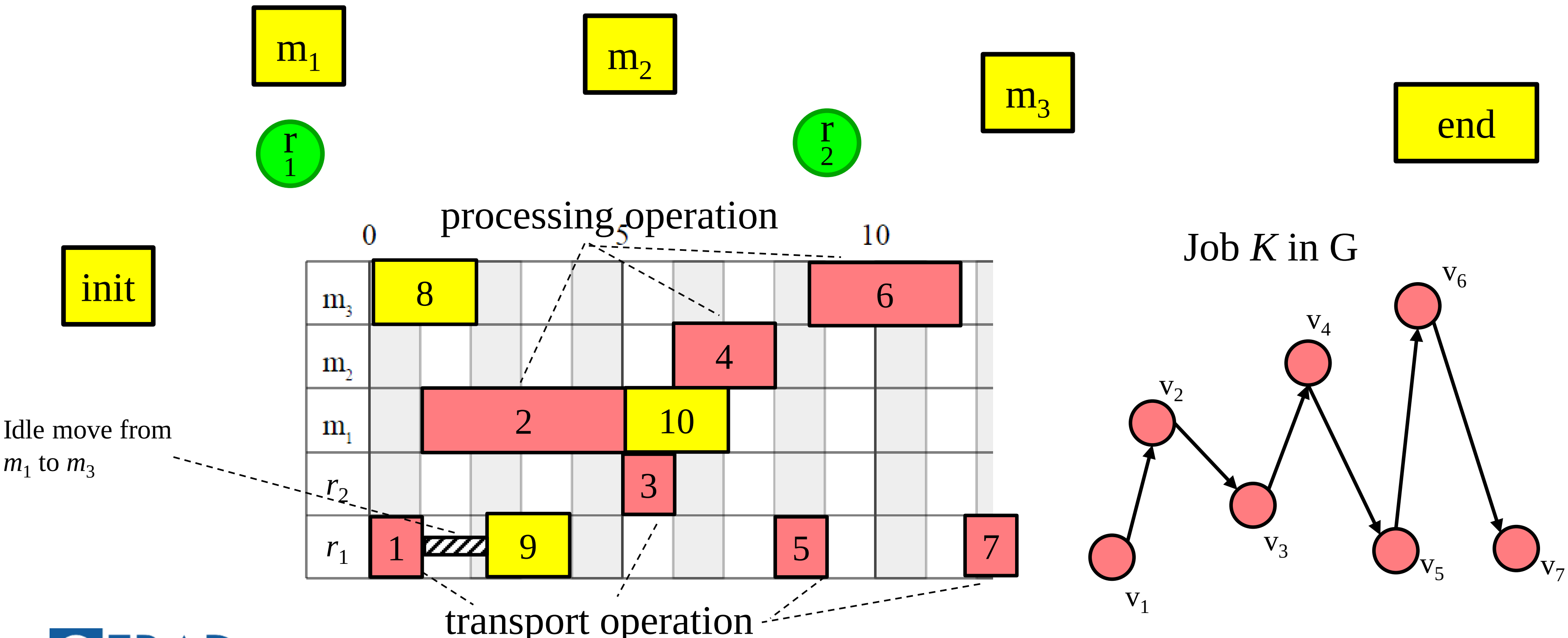
If $k_i = 0$ then it is a no-wait constraint

$$x_j - x_i \le p_i + k_i \iff$$
$$x_i - x_j \ge -(p_i + k_i)$$



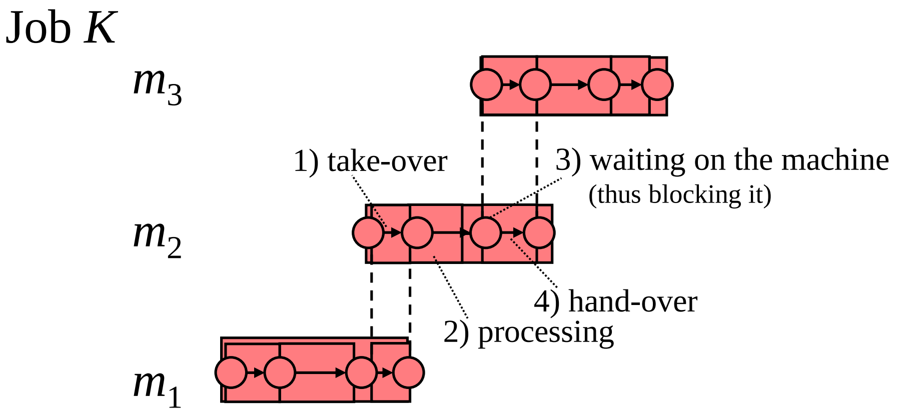
Transportation by Mobile Devices

- Mobile devices transport the jobs between the machines
- If the mobile devices can hold at most one job at any time, model them as machines
- Use sequence-dependent setup times for idle moves.



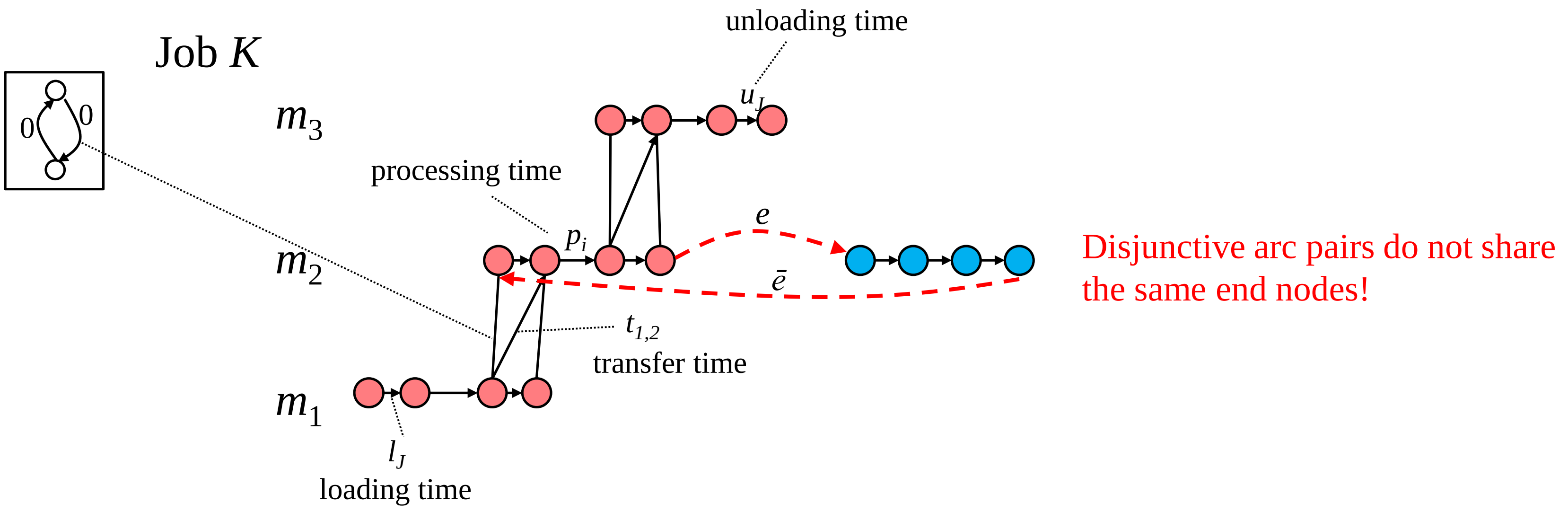
No (Intermediate) Storage Space - Blocking

- In the classical job shop, storage is not considered
  - Assumption: jobs can be stored somewhere
- However, often the storage space is limited or no (intermediate) storage space at all (e.g. robotic cells)
- Assume no storage space available, so called blocking.



No (Intermediate) Storage Space - Blocking

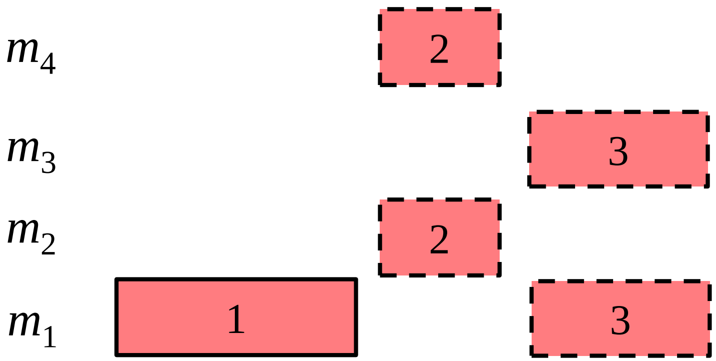
- In the classical job shop, storage is not considered
  - Assumption: jobs can be stored somewhere
- However, often the storage space is limited or no (intermediate) storage space at all (e.g. robotic cells)
- Assume no storage space available, so called blocking.

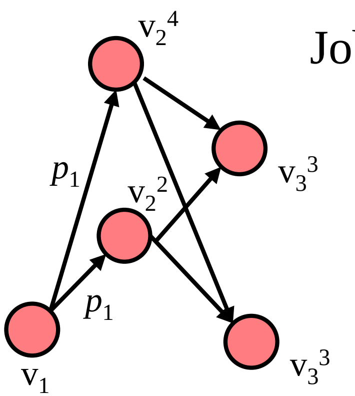


- Small number of storage units available: model them as machines

Routing Flexibility

- Different types of routing flexibility
  - Kis, T. (2003). Job-shop scheduling with processing alternatives. *European Journal of Operational Research*, *151*(2), 307–332
- Consider independent choice of the machine for each operation
  - For each operation i, the machine of i is not fixed but can be chosen from a subset of machines $M_i \subseteq M$
  - (Variable) Mode μ: choice of a machine $\mu_i \in M_i$ for each operation $i \in I$





Job K in G

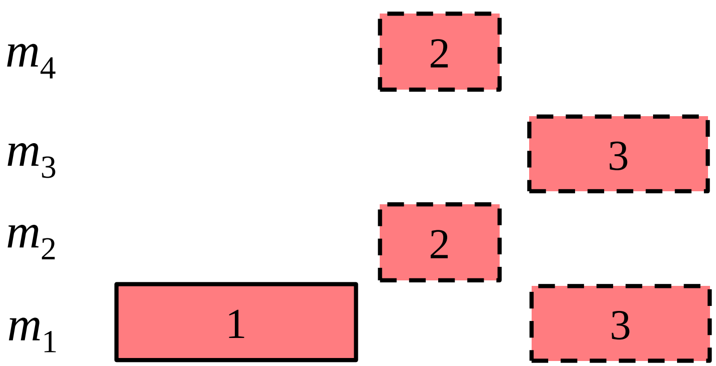
- With given mode μ, disjunctive graph $G^{\mu} = (V^{\mu}, A^{\mu}, E^{\mu}, \mathcal{E}^{\mu}, d)$
  (node-induced subgraph of G where nodes not belonging to mode μ are deleted)

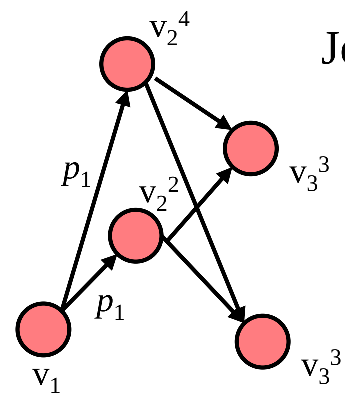
- Extended definition of selections:

Definition
A selection (μ,S): any mode μ and set $S \subseteq E$ of disjunctive arcs
A selection S is complete if $S \cap \{e,\bar{e}\} \neq \emptyset$ for all $\{e,\bar{e}\} \in \mathcal{E}^{\mu}$
A selection S is positive acyclic if the graph $G(\mu,S)=(V^{\mu},A^{\mu} \cup S,d)$ has no cycle of positive length and positive cyclic otherwise.

Among all feasible selections, find a selection (μ,S) minimizing the length of a longest path from σ to τ.

Gröflin, H., Pham, D. N., & Bürgy, R. (2011). The flexible blocking job shop with transfer and set-up times. *Journal of Combinatorial Optimization*, *22*(2), 121–144.

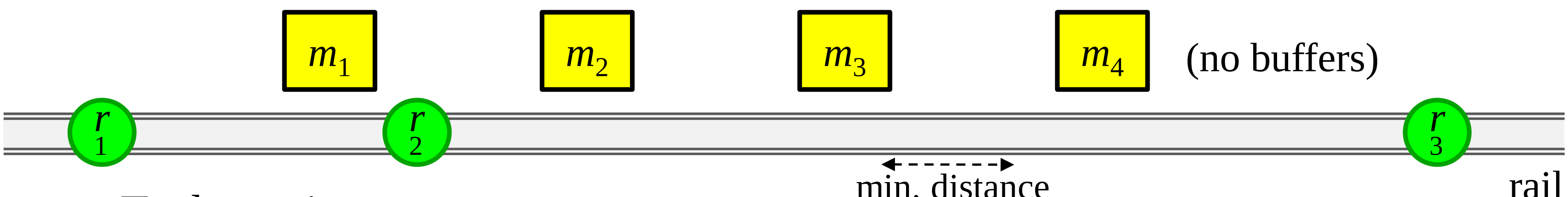




Job K in G

2.2. An Application: The BJS-RT and ALPHABOT
The BJS-RT

- Version of the blocking job shop (with routing flexibility) characterized by
- a rail-bound transportation system consisting of mobile devices (robots, cranes, …)
  - Processing of jobs on machines
  - Transport from one machine to the next by a robot which can be chosen
  - Robots move on a rail along which the machines are located
  - Robots cannot pass each other, maintain a minimal distance from each other
  - Each robot can handle at most one job at any time
  - Each robot can move at a speed up to a (robot-dependent) speed limit

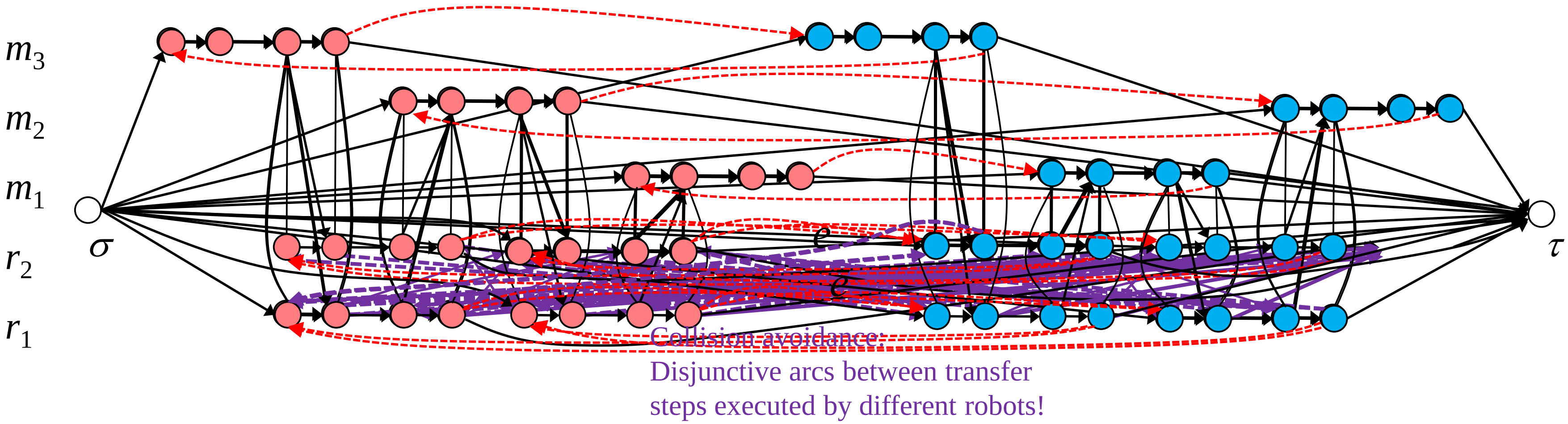


- To determine
  - Machine (robot) assignments and starting times (for processing and transport op.)
  - But also feasible trajectories of the robots

typical scheduling variables

No scheduling variables (and not in objective function)

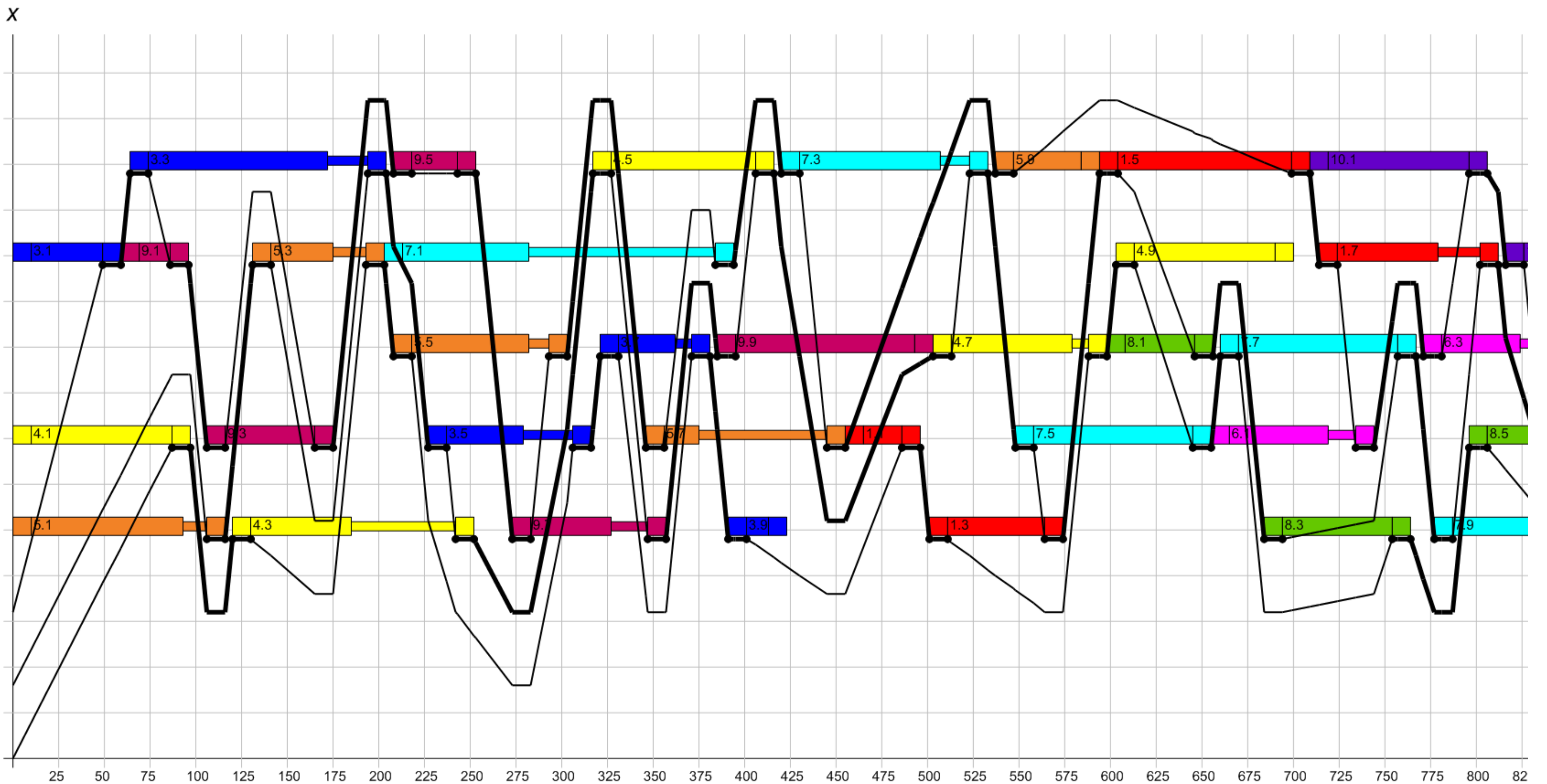
- Projection of the solution space onto the space of the assignment and time variables
  - Adding disjunctive arc pairs between *transfer steps* executed *by different robots*!



  - Yielding a formulation of the BJS-RT in a disjunctive graph
  - Allowing to apply our solution approach
  - Establish efficient algorithms for the feasible trajectory problem

See: Bürgy, R., & Gröflin, H. (2016). The blocking job shop with rail-bound transportation. *Journal of Combinatorial Optimization*, *31*(1), 151–181.

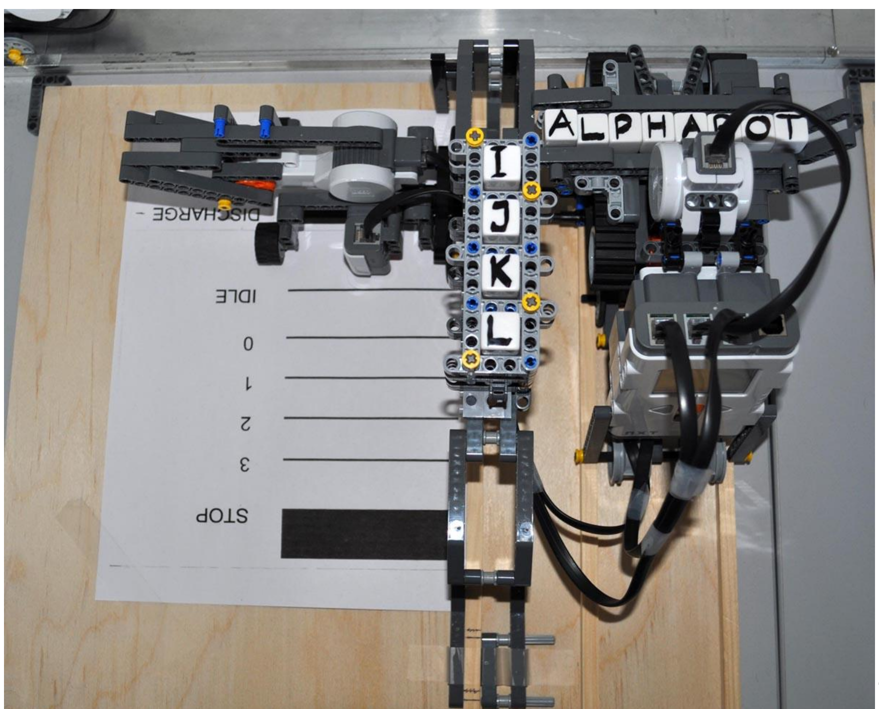
BJS-RT Schedules

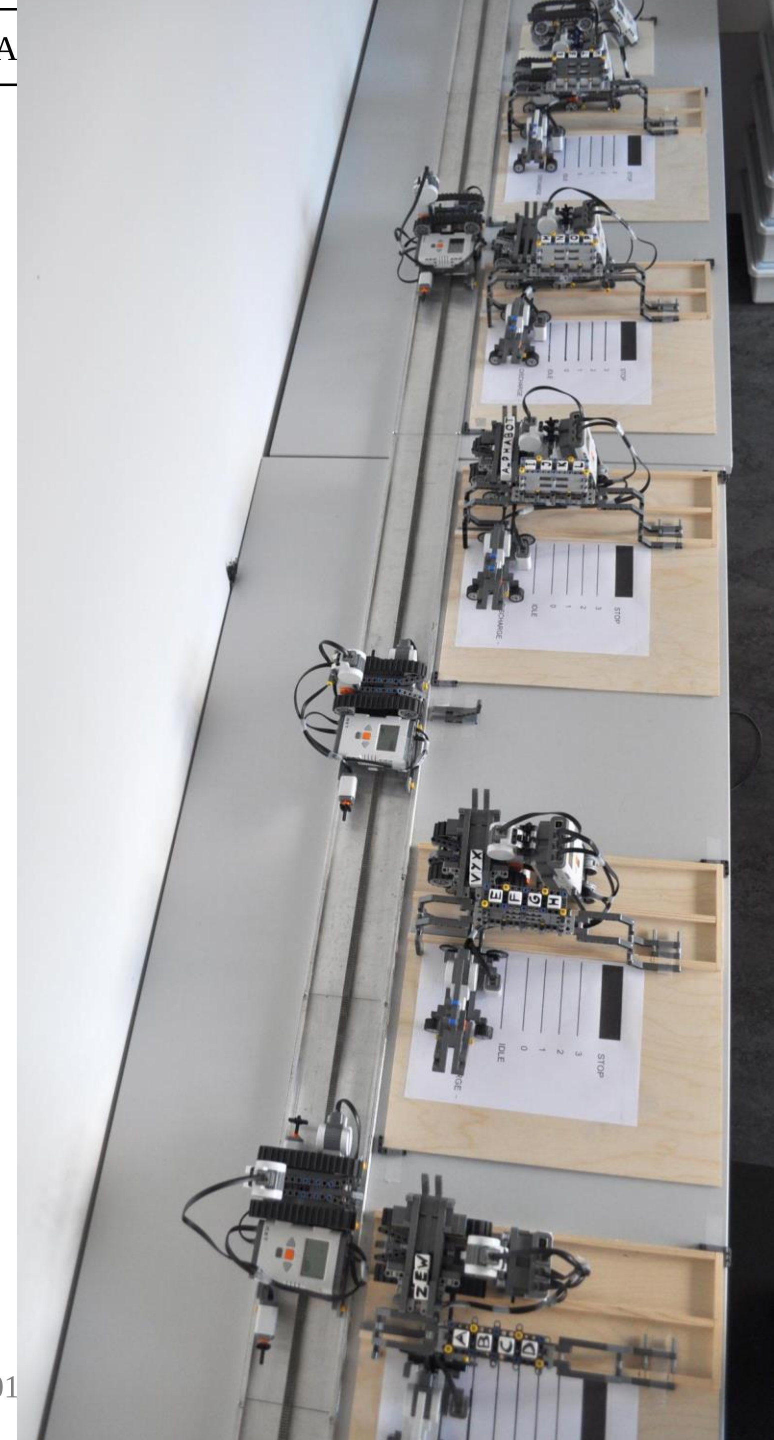


A “location-extended” Gantt chart

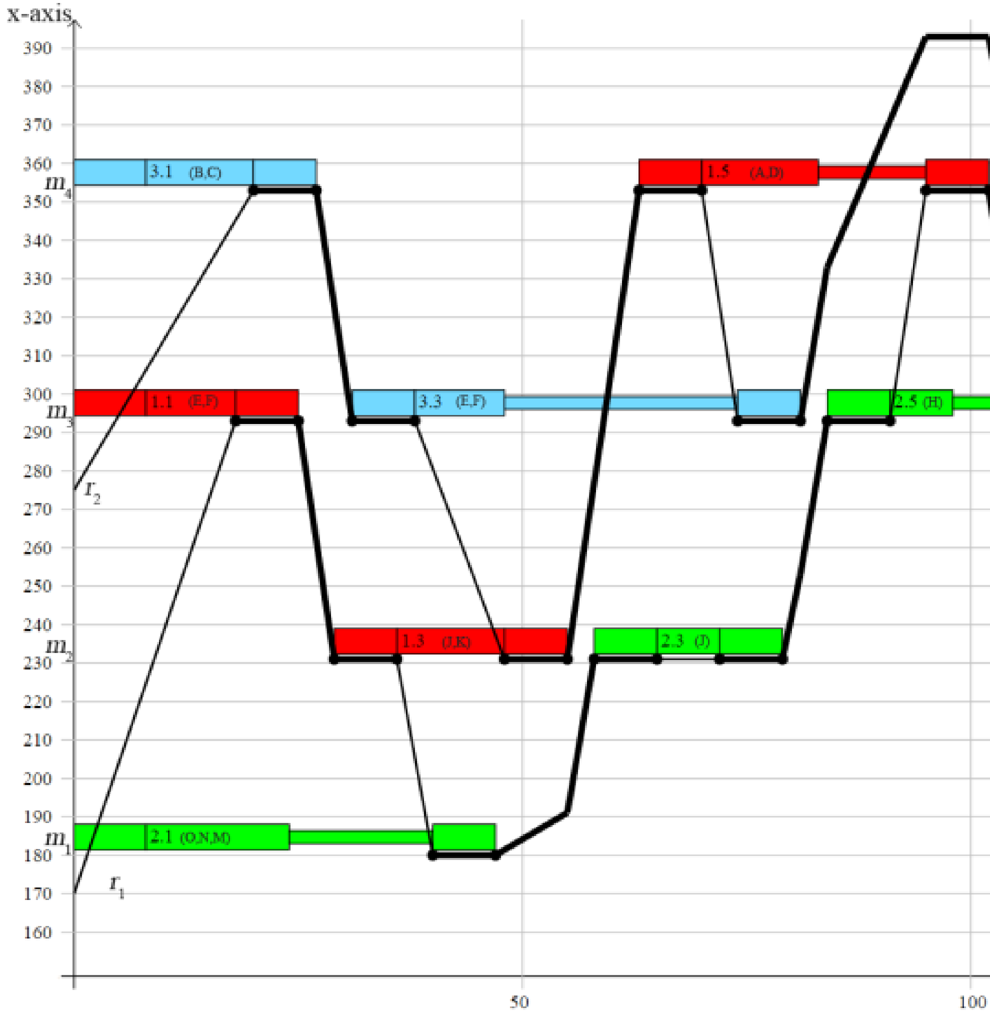
The ALPHABOT

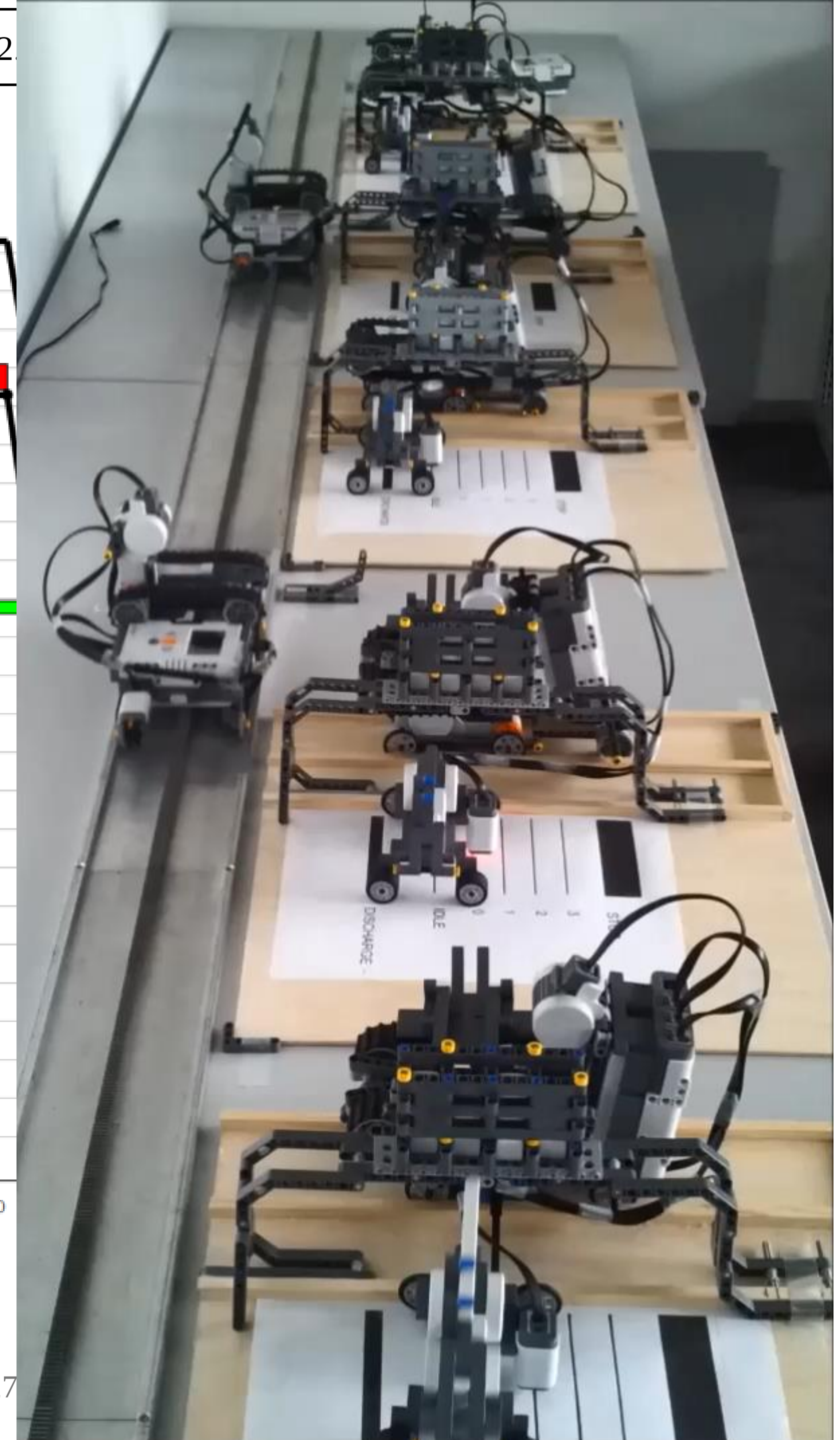
- A physical model of the BJS-RT
  - Machine: contains stacks of dices (with same letter)
  - Assemble words (a small container holds the dices)
  - Produce a given set of words as fast as possible





ALPHABOT in Action



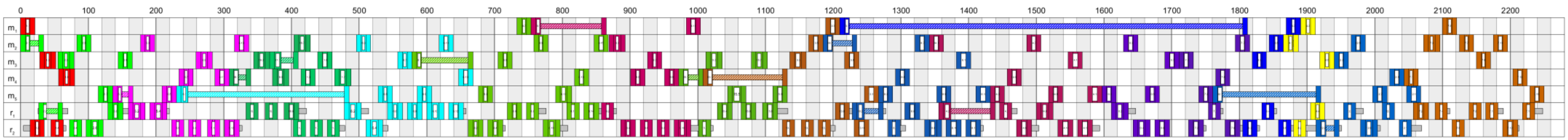


Robots and machines controlled by precedence constraints and NOT by (fixing) times!

Video: http://research.reinhardbuergy.ch/alphabot.php

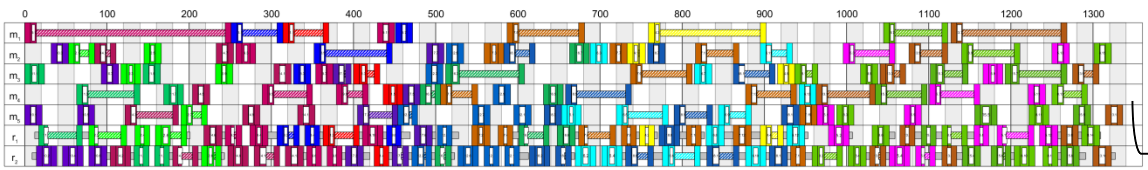
The Value of Optimization

- Example: Instance with 18 words (names)
  - Simple solution (job permutation schedule):



(trajectories omitted)

  - An optimized solution:



  - Nontrivial (even impossible) to find good plans by hand

improvement of 40%

- Some benefits
  - Quality improvement
  - (Semi-) automating complex work (decision support)
  - The role of the planner changes (addressing tactical questions)
  - Increased flexibility in the planning task

Overview

1. The Classical Job Shop Scheduling Problem
   1. Introduction
   2. A Combinatorial Formulation in a Disjunctive Graph
   3. Applications in Practice
2. Complex Process Features
   1. Some Process Features
   2. An Application: The BJS-RT and ALPHABOT
3. Complicated Objectives
   1. General Regular Objective
   2. A Class of Convex Cost Objectives
4. A Local Search Solution Approach
   1. The Job Insertion Problem
   2. Local Moves and Locally Improving Moves
   3. Some Computational Results

3. Complicated Objectives

- Consider again the timing problem in the classical job shop
- Timing subproblem:
  Given a selection S, solve:

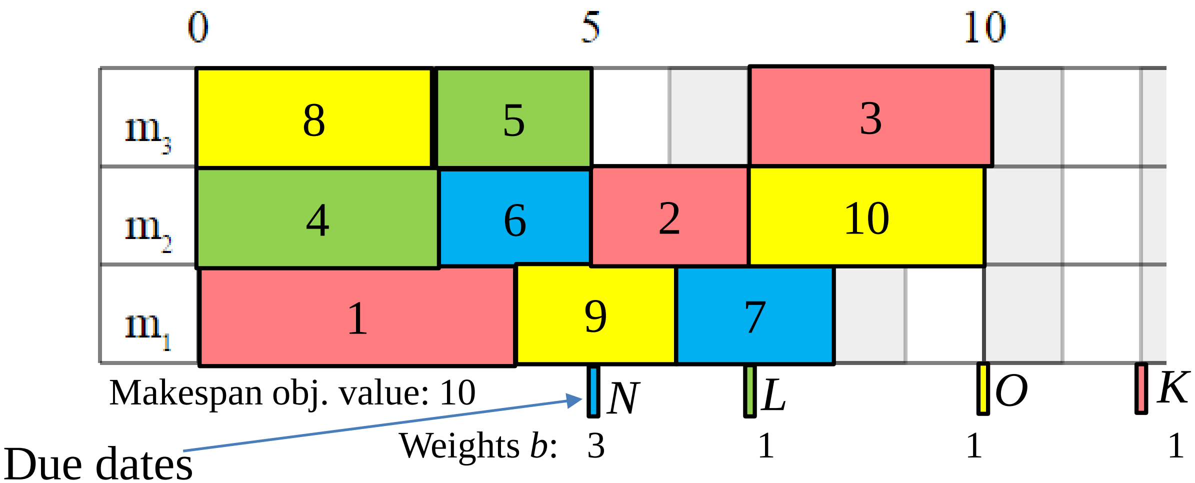
$$\begin{aligned}&\text{minimize} \quad \alpha_\tau - \alpha_\sigma \\ &\text{subject to}: \\ &\alpha_w - \alpha_v \geq d_{vw} \quad \text{for all } (v,w) \in A \cup S\end{aligned}$$

- Its dual, is a (simple) network flow problem.
- How to solve it: an optimal solution can be found by
  - Computing the earliest time schedule $\alpha(S)$ where
  - For all $v \in V$, $\alpha_v(S)$: length of a longest path from σ to node v in $G(S)$

3.1. General Regular Objective

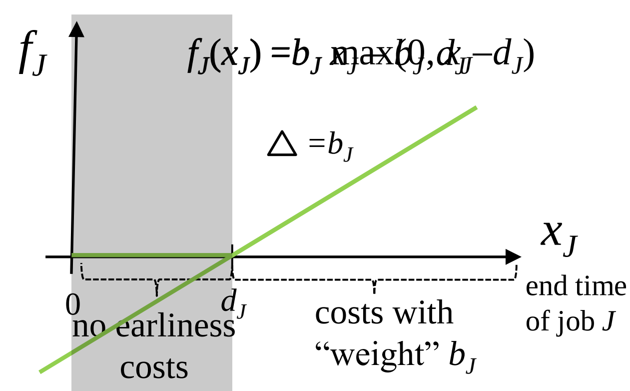
- It this procedure just applicable to the makespan objective?
- The class of regular objectives
  - "the earlier the better"
  - Formally, a function $f: \Re^V \rightarrow \Re$ is called regular if for all $\alpha, \alpha' \in \Re^V$, $\alpha \leq \alpha' \Rightarrow f(\alpha) \leq f(\alpha')$
  - Comprises many objectives: makespan, total flow time, total (weighted) tardiness, total squared tardiness, etc.
- The earliest time schedule $\alpha(S)$ is optimal
  - Let $\Omega(S)$ be the solution space of the timing problem
  - Clearly, $\alpha(S) \leq \alpha$ for all $\alpha \in \Omega(S)$, implying $f(\alpha(S)) \leq f(\alpha')$
  - Timing problem efficiently solvable!
  - Similar computational effort as for makespan objective
  - Somewhat higher for "re-optimization"
    - Consider: Vidal, T., Crainic, T. G., Gendreau, M., & Prins, C. (2015). Timing Problems and Algorithms: Time Decisions for Sequences of Activities. Networks, 65(2), 102–128.

An Example with Tardiness Costs

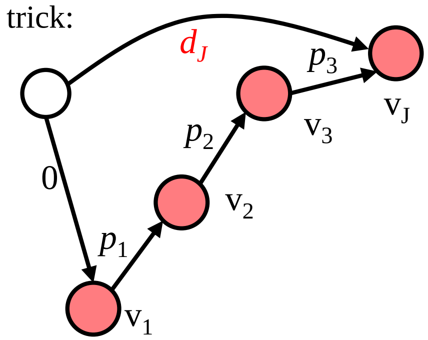


Total linear tardiness costs: 9 (job N 3 time units too late)

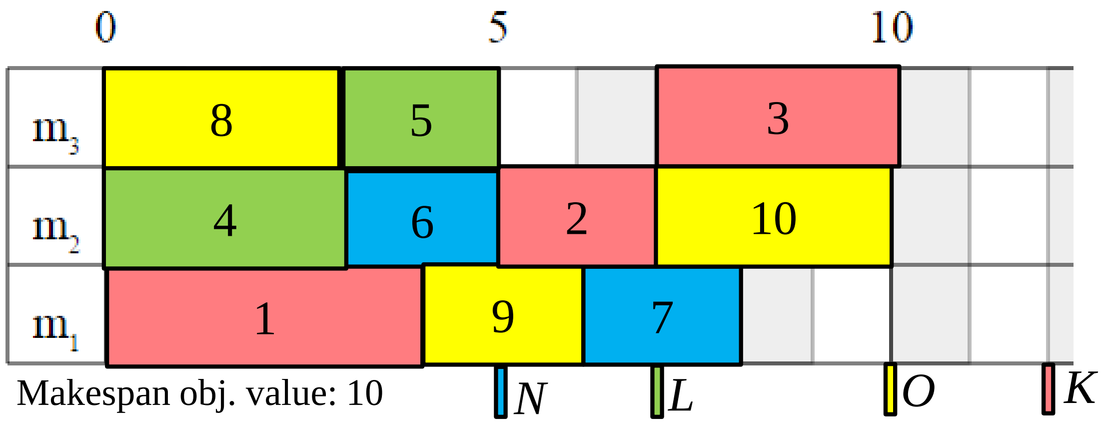
Linear tardiness costs (for each job J)



Note: non-linear function in x_J, simple "linearization" modeling trick:

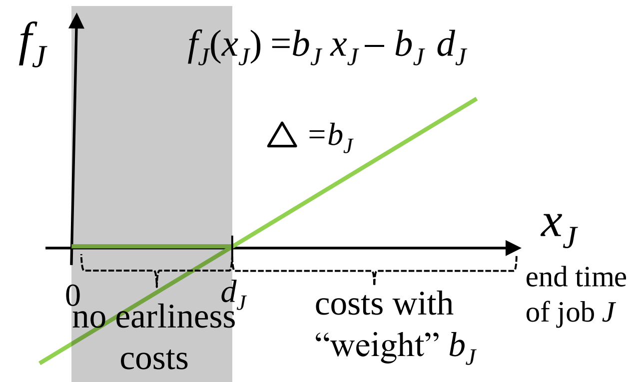


An Example with Tardiness Costs



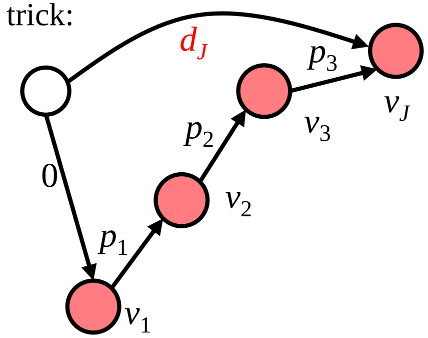
Weights b: 3 1 1 1

Linear tardiness costs (for each job J)

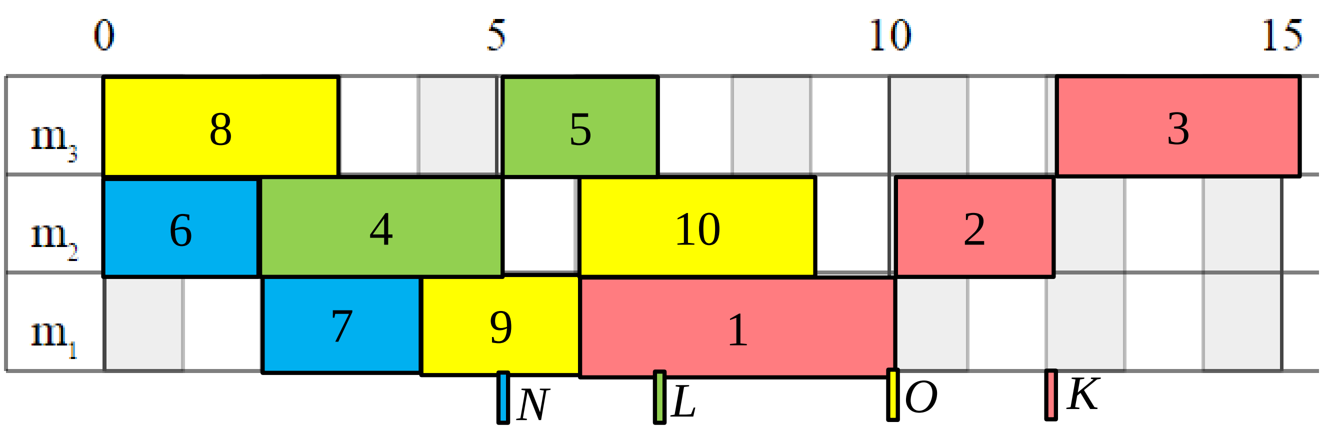


Total linear tardiness costs: 9 (job N 3 time units too late)

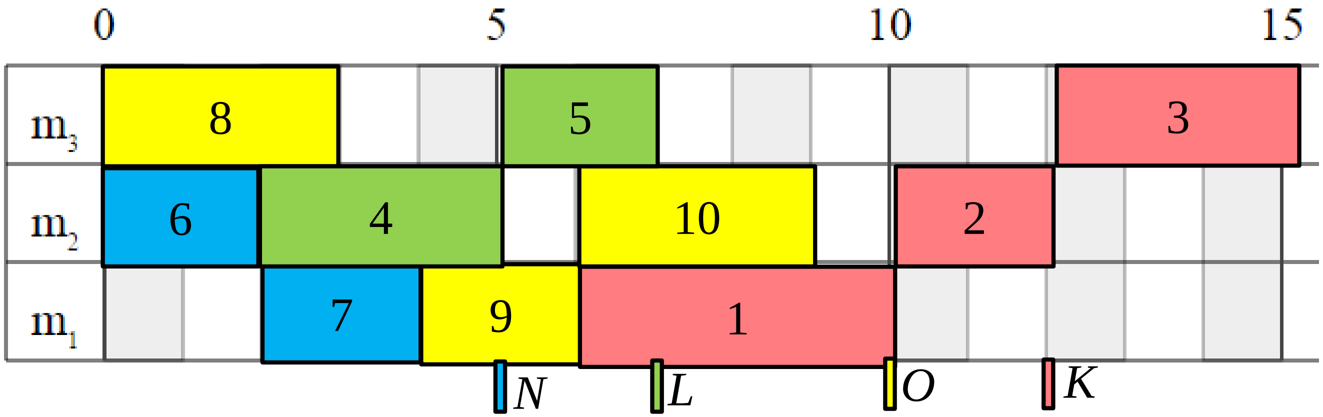
Note: non-linear function in x_J, simple "linearization" modeling trick:



With improved sequencing:



Total linear tardiness costs: 3 (job K 3 time units too late)

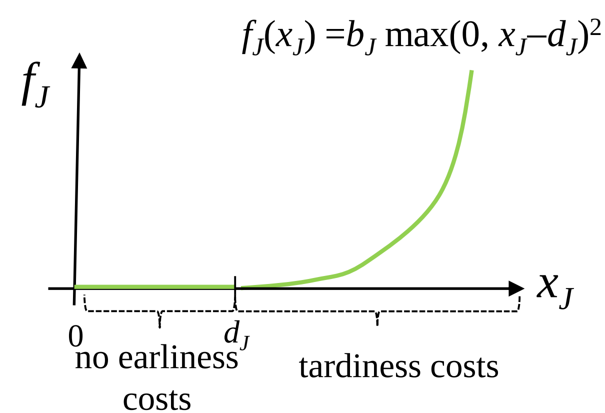


Total linear tardiness costs: 3 (job K 3 time units too late)

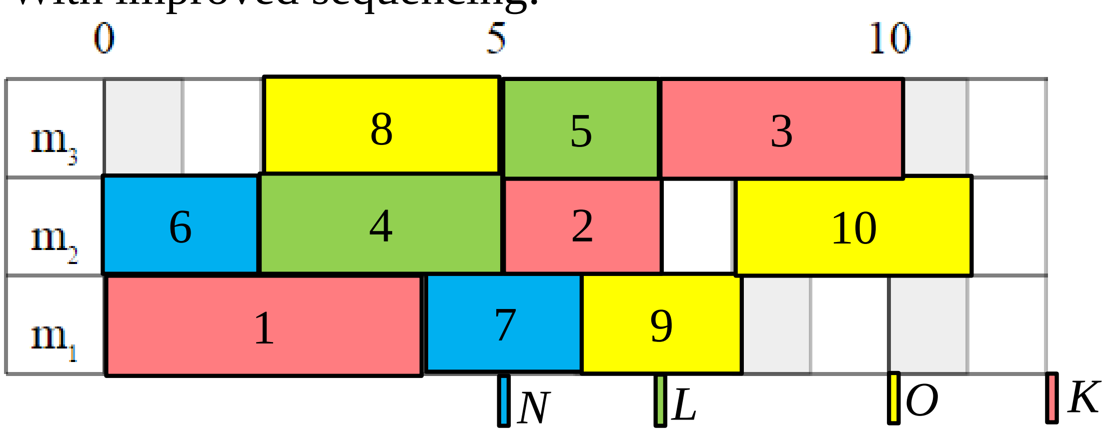
Total squared tardiness costs: 9 (job K 3 time units too late)

However, a large tardiness may be very undesirable in practice → squared tardiness costs:

Squared tardiness costs



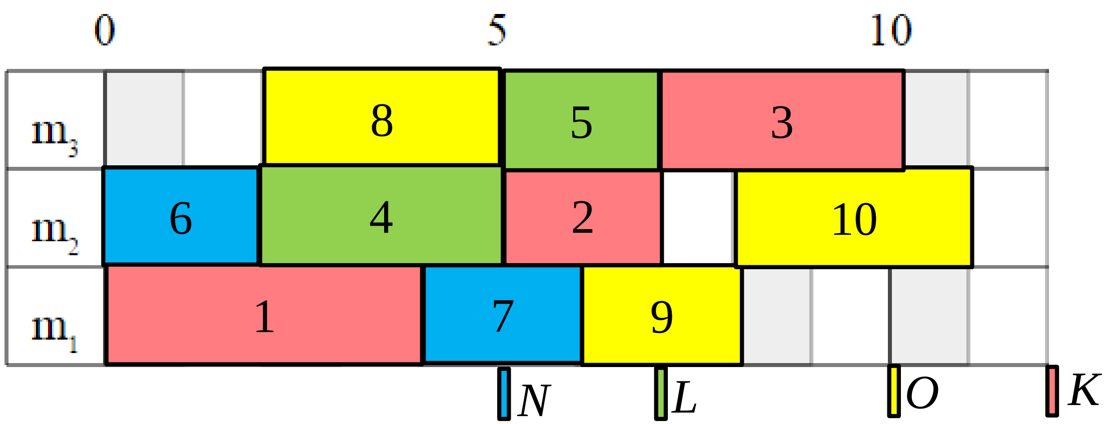
With improved sequencing:

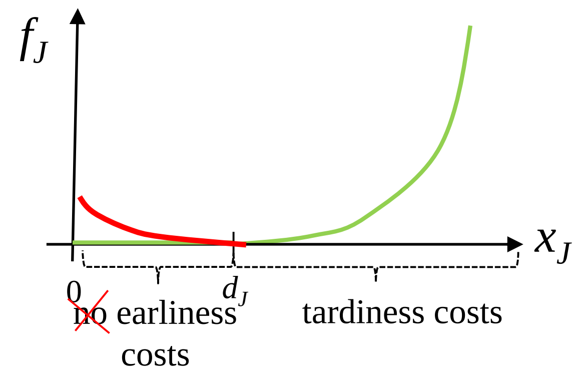


still regular objective!

Total squared tardiness costs: 4 (Job O 1 and Job N 1 time unit too late)

3.2. A Class of Convex Cost Objectives

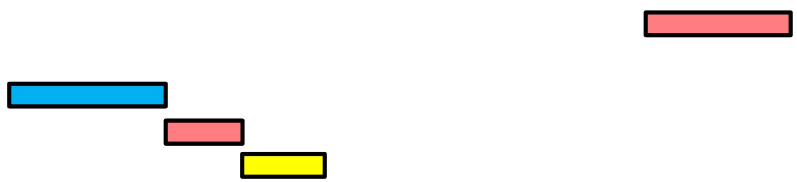




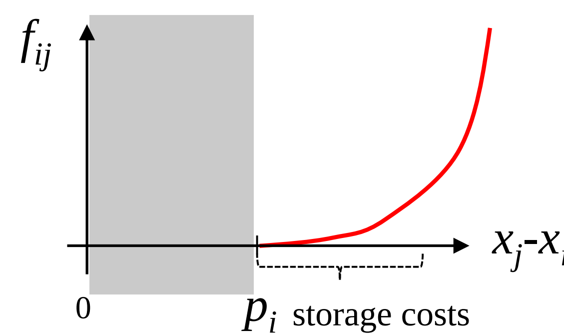
Total squared tardiness costs: 4 (Job *O* 1 and Job *N* 1 time unit too late)

Earliness:

Storage:



Op. *i* and *j* consecutive in some job



Just-in-time scheduling:
Take into account convex tardiness, earliness and storage costs!

Total Convex Costs: The Timing Problem

Makespan objective:

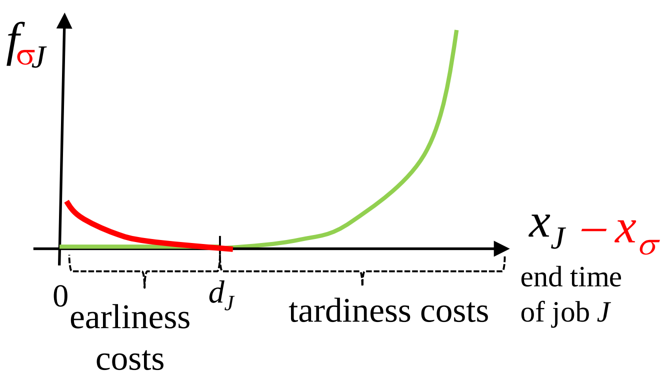
$$\begin{aligned}&\text{minimize} \quad \alpha_\tau - \alpha_\sigma\\&\text{subject to:}\\&\alpha_w - \alpha_v \geq d_{vw} \quad \text{for all } (v,w) \in A \cup S\end{aligned}$$

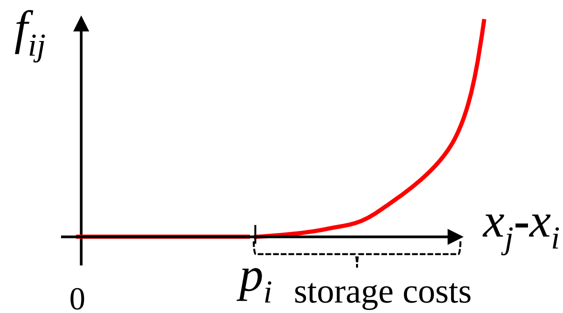
A total convex costs objective:

$$\begin{aligned}&\boxed{\text{minimize} \quad \sum_{(v,w)\in F} f_{vw}(\alpha_w - \alpha_v)} \quad \boxed{f_{vw} \text{ convex}}\\&\text{subject to:}\\&\alpha_w - \alpha_v \geq d_{vw} \quad \text{for all } (v,w) \in A \cup S\end{aligned}$$

"Precision" of our plan.
(implied with regular objective and integer data)

U: "large" planning horizon
(for technical reasons)





A Solution Approach for This Timing Problem

f_{vw} convex

$$\text{minimize} \quad \sum_{(v,w)\in F} f_{vw}(\alpha_w - \alpha_v)$$

subject to:

$$\alpha_w - \alpha_v \ge d_{vw} \quad \text{for all } (v,w) \in A \cup S$$

$$\alpha_v \text{ integer for all } v \in V$$

$$0 \le \alpha_v \le U \quad \text{for all } v \in V$$

- Still a network flow problem?
- A convex cost integer dual network flow problem!

Ahuja, R. K., Hochbaum, D. S., & Orlin, J. B. (2003). Solving the Convex Cost Integer Dual Network Flow Problem. *Management Science*, *49*, 950–964

  - Show that the Lagrangian relaxation of the problem (actually a reformulation) can be transformed to a network flow problem with (special) convex costs
  - Adapt the cost-scaling algorithm for the minimum cost flow problem to solve the convex cost network flow problem (obtaining also an optimal dual solution)
  - Overall time complexity: $O(nm \log(n^2/m) \log(nU))$

also shown by:

- Hence, timing problem still efficiently solvable!
  - Higher time complexity than for regular objectives

Stephan Foldes and François Soumis. PERT and crashing revisited: Mathematical generalizations. *European Journal of Operational Research* 64.2 (1993): 286-294.

GERAD

Overview

1. The Classical Job Shop Scheduling Problem

1. Introduction
2. A Combinatorial Formulation in a Disjunctive Graph
3. Applications in Practice

2. Complex Process Features

1. Some Process Features
2. An Application: The BJS-RT and ALPHABOT

3. Complicated Objectives

1. General Regular Objective
2. A Class of Convex Cost Objectives

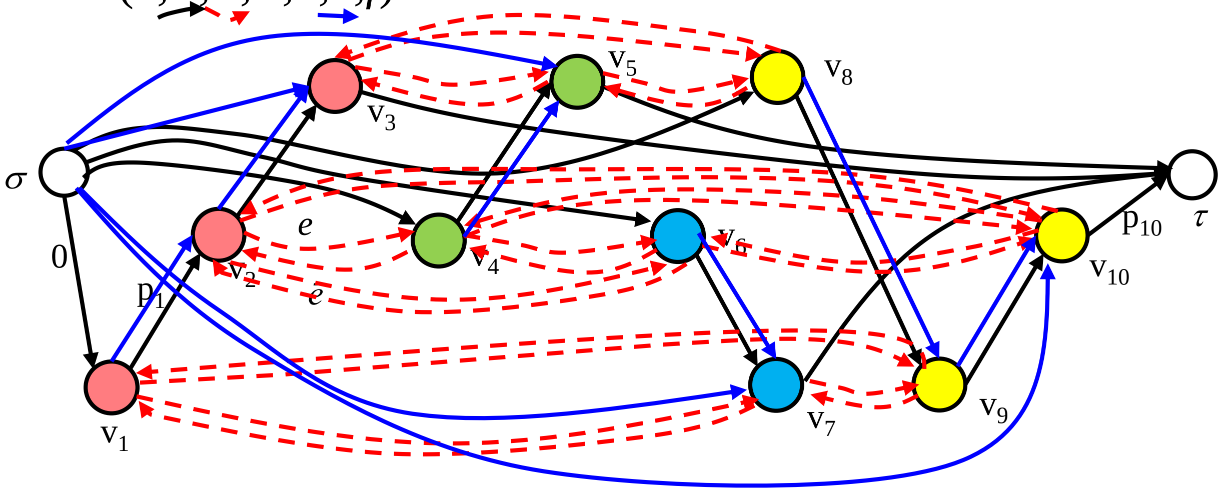
4. A Local Search Solution Approach

1. The Job Insertion Problem
2. Local Moves and Locally Improving Moves
3. Some Computational Results

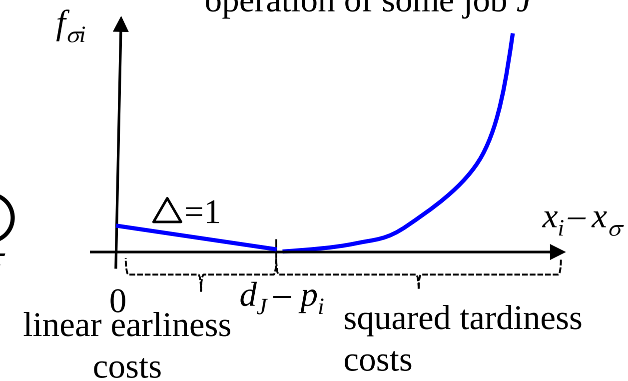
4. A Local Search Solution Approach

- The job shop scheduling problem with convex costs:

$G=(V,A,E,\mathcal{E},d,F,f)$



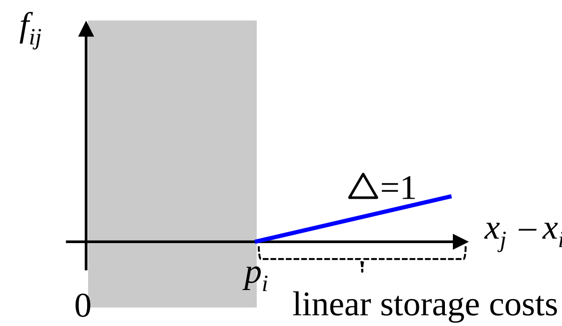
Op. i being the last operation of some job J



$f_{\sigma i}(x_i - x_\sigma) = \max(0, (d_J - p_i) - (x_i - x_\sigma)) + \max(0, (x_i - x_\sigma) - (d_J - p_i))^2$

Just-in-time job shop scheduling with squared tardiness costs and linear storage costs

Op. i and j consecutive in some job

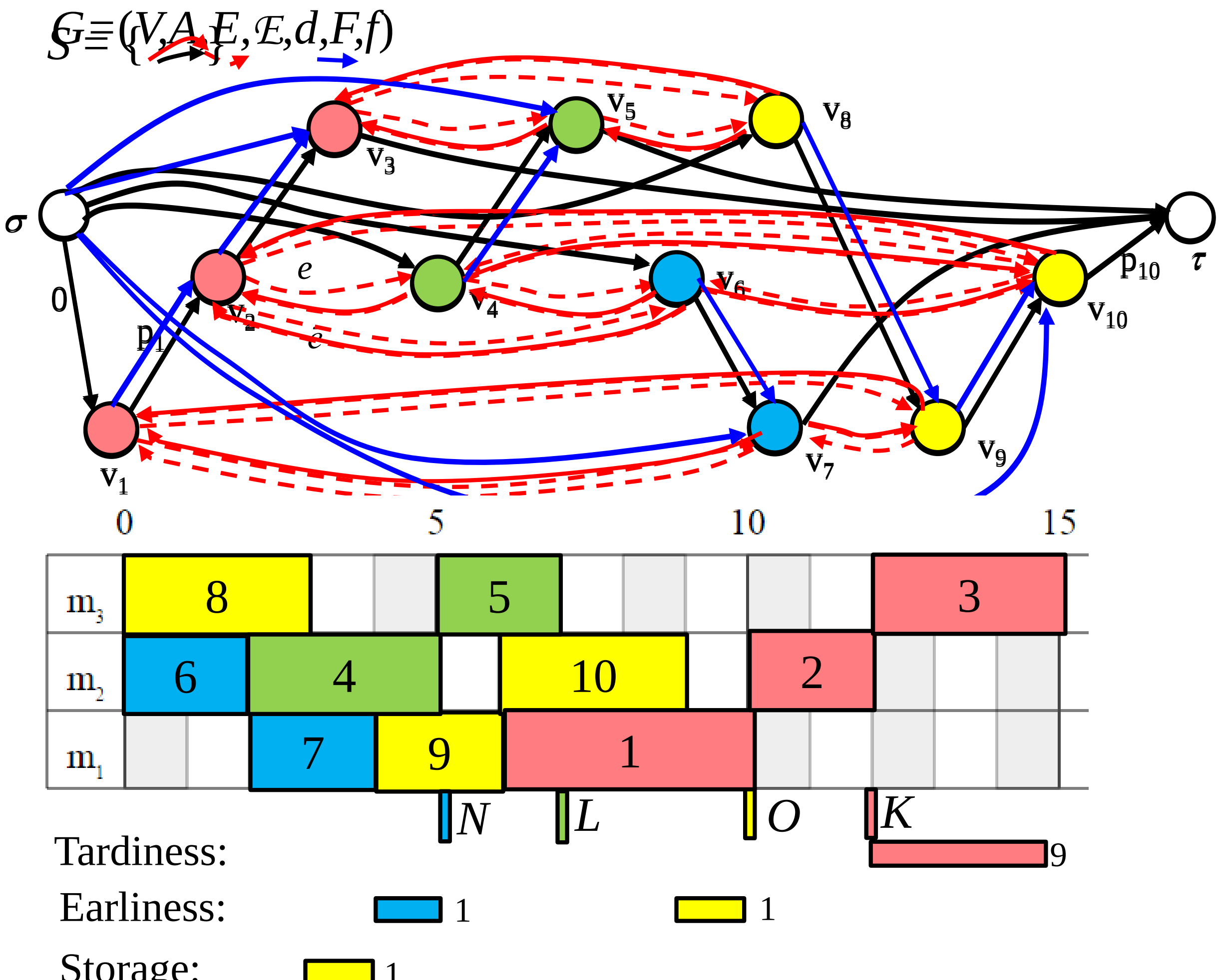


Among all feasible selections, find a selection S with minimum total convex costs.

$f_{\sigma i}(x_j - x_i) = x_j - x_i - p_i$

General Scheme

- Local search based on a job insertion neighborhood



Total costs: 12

Neighbor generation: Extract a job and re-insert it into the given schedule

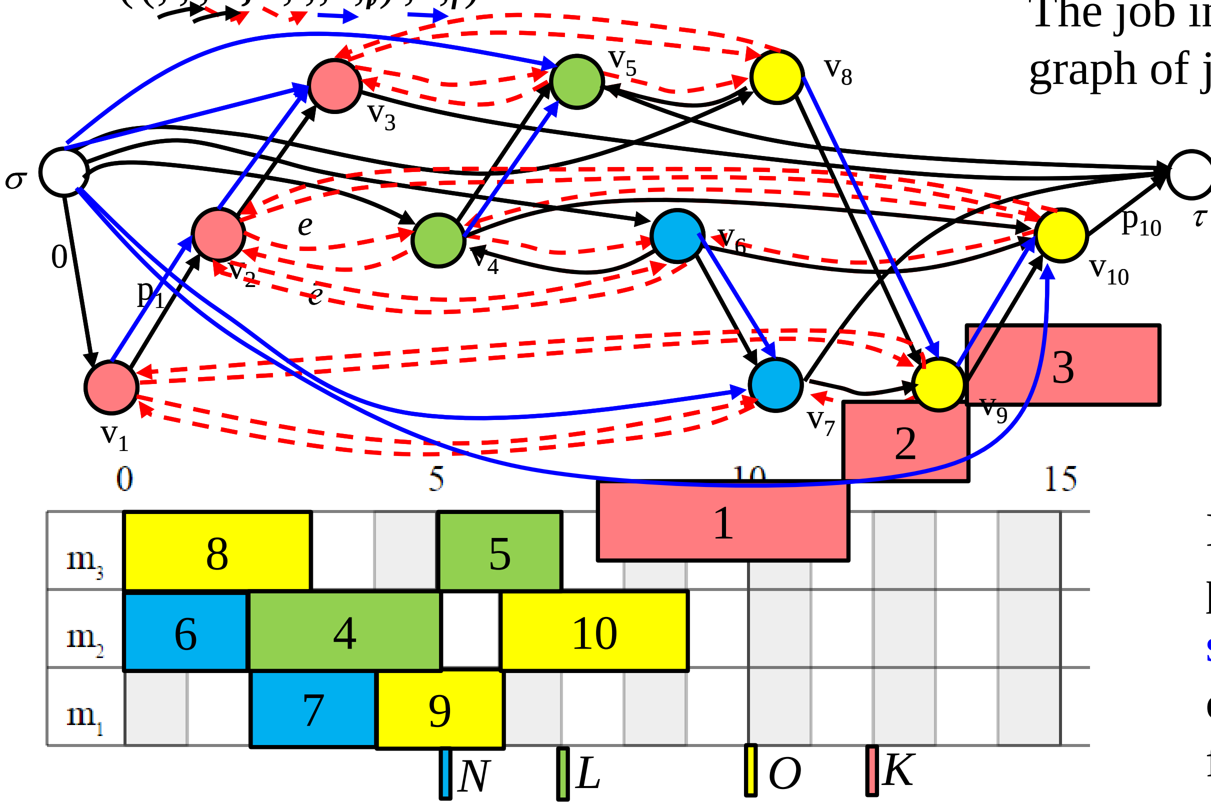
„Given schedule“: fixing disjunctive arcs, not starting times!

See:
Gröflin, H., & Klinkert, A. (2007). Feasible insertions in job shop scheduling, short cycles and stable sets. *European Journal of Operational Research*, *177*(2)

4.1. The Job Insertion Problem

- Problem formulation in its associated disjunctive graph:

$G^K=(V,A^K,E^K,d,F^K,f)$

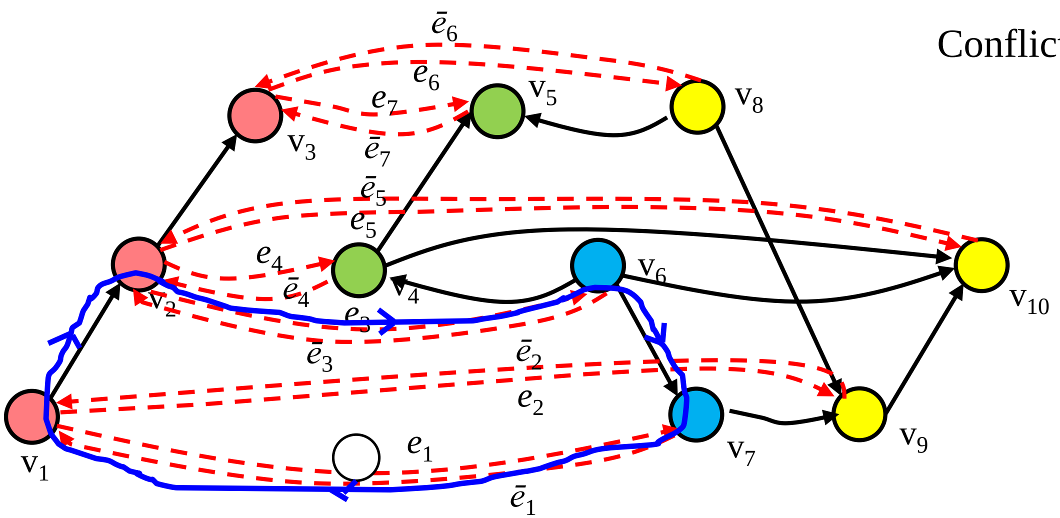


The job insertion disjunctive graph of job *K*.

In this graph, (complete, positive acyclic, feasible) selections (complete are called positive acyclic, feasible) insertions

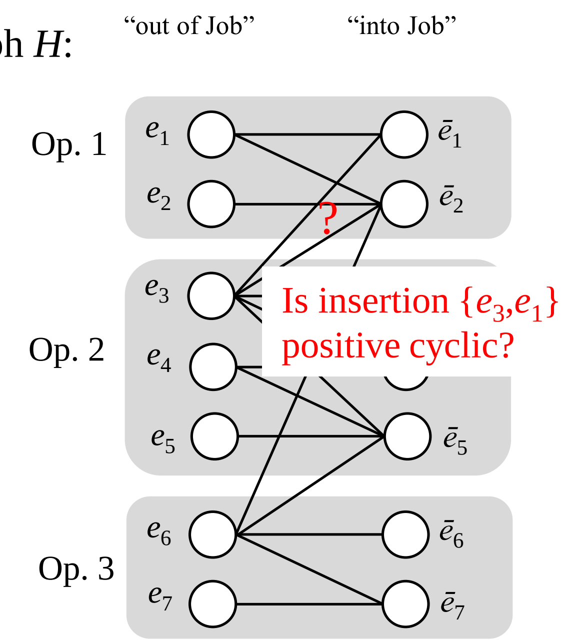
The Conflict Graph

- Characterize ALL feasible insertions in an associated graph



(nodes σ, τ and arcs of F omitted)

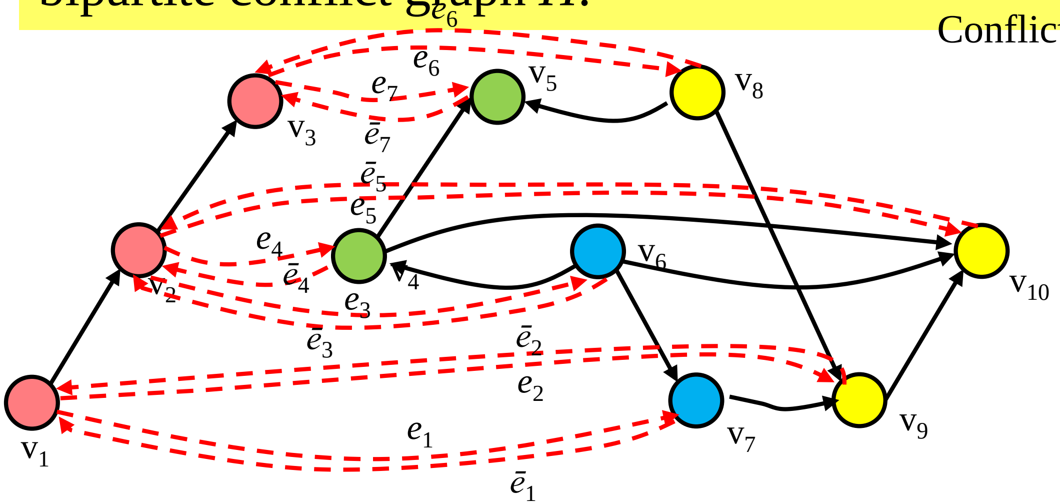
Conflict Graph H:



Definition
Given a job insertion graph $G^J = (V^J,A^J,E^J,\mathcal{E}^J,d,F,f)$, the conflict graph of G^J is the undirected graph $H= (E^J,U)$ where for any $e,f \in E^J$, $(e,f\,) \in U$ if insertion $\{e,f\}$ is positive cyclic.

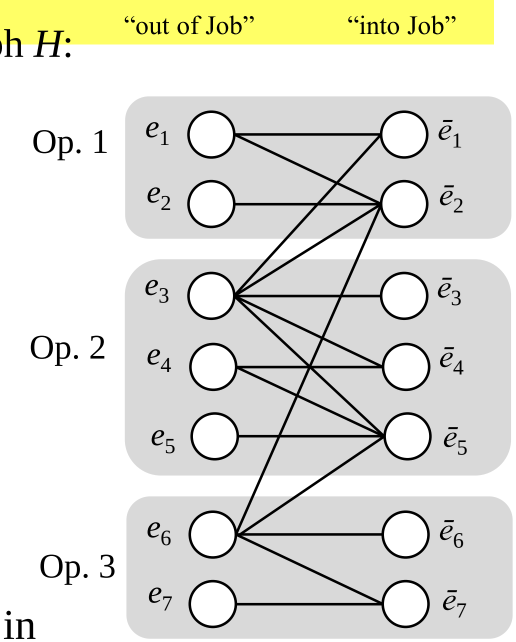
Theorem

Given a job insertion graph $G^J = (V^J, A^J, E^J, \mathcal{E}^J, d)$, the feasible insertions are in one-to-one correspondence with the stable sets of size $|E^J|/2$ in the bipartite conflict graph H.



(nodes σ, τ and arcs of F omitted)

Conflict Graph H:

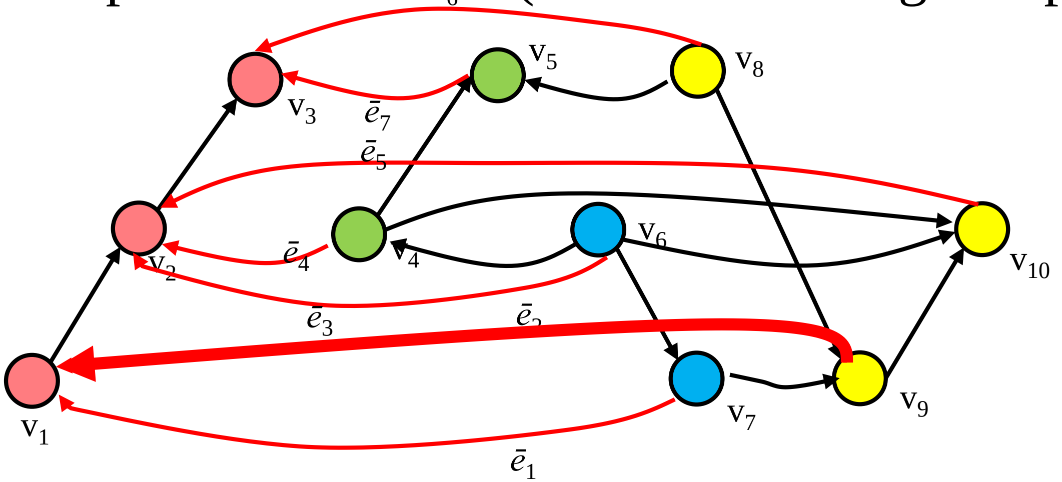


Proof, see: Gröflin, H., & Klinkert, A. (2007). Feasible insertions in job shop scheduling, short cycles and stable sets. *European Journal of Operational Research*, *177*(2)

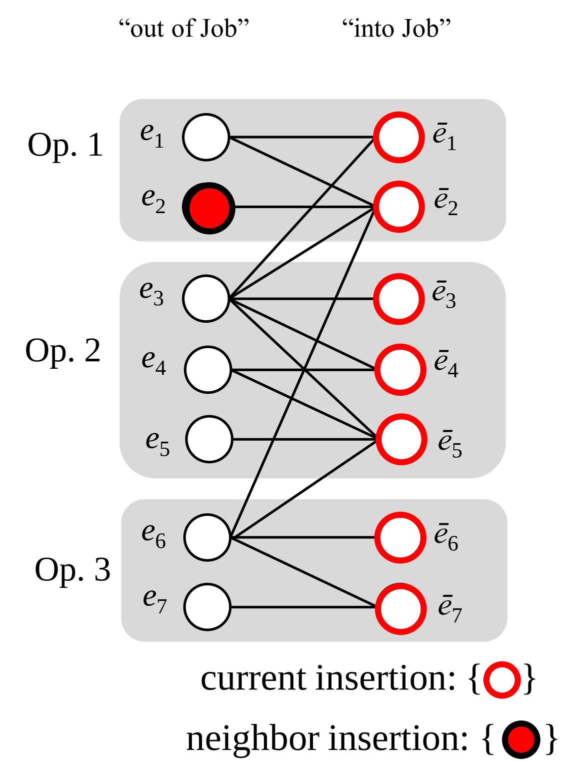
- $\Rightarrow$ Nice (polyhedral) characterization of all feasible insertions
- Generate neighbor insertions in the conflict graph.

4.2. Local Moves and Locally Improving Moves

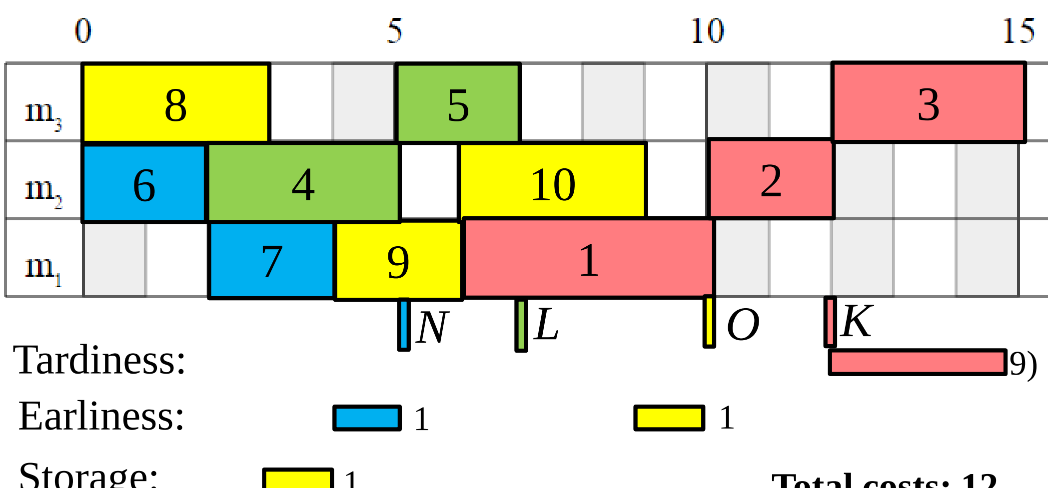
- Replace a “critical” disjunctive arc, i.e. a disjunctive arc with positive arc flow (in dual of timing sub-problem), by its mate



Conflict Graph H:

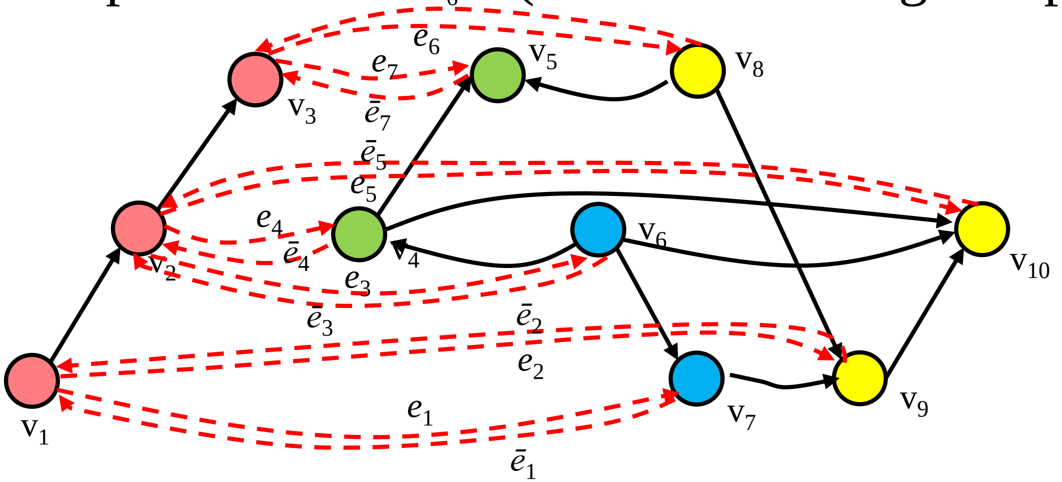


Arc flow of $\bar{e}_2$ is 2. Replace $\bar{e}_2$ by e_2.

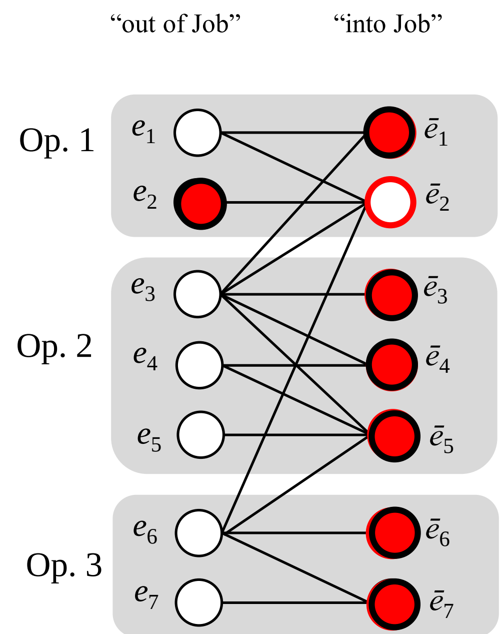


4.2. Local Moves and Locally Improving Moves

- Replace a "critical" disjunctive arc, i.e. a disjunctive arc with positive arc flow (in dual of timing sub-problem), by its mate



Conflict Graph H:



current insertion: { }

neighbor insertion: { }

Arc flow of $\bar{e}_2$ is 2. Replace $\bar{e}_2$ by e_2.

- Generate nearest insertion:

$$T_{\bar{e}} = \bar{e} \cup (T^S \setminus \{e\})$$

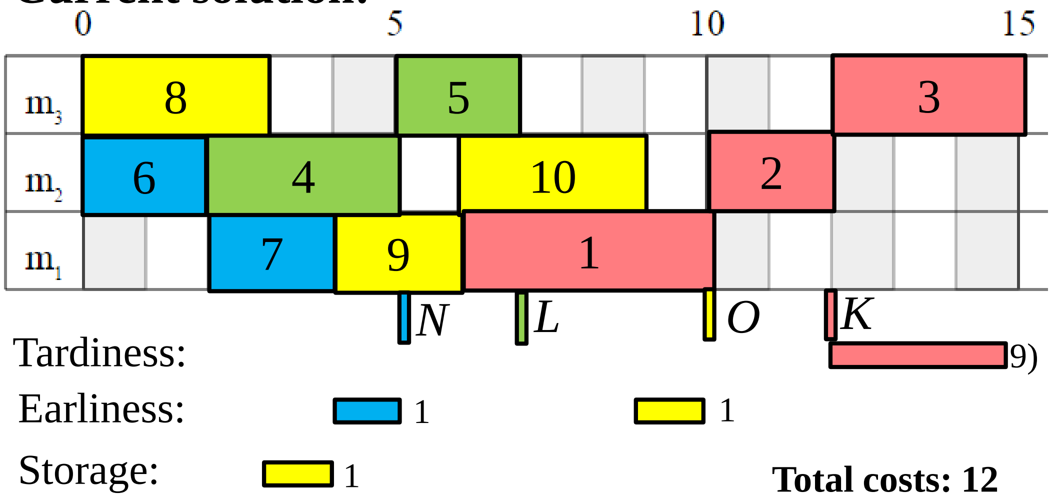
"Swapping" a critical arc!

Proposition
$T_{\bar{e}}$ is a feasible insertion.

See e.g. Brandimarte, P., & Maiocco, M. (1999). Job shop scheduling with a non-regular objective: A comparison of neighbourhood structures based on a sequencing/timing decomposition. *International Journal of Production Research*, *37*(8), 1697–1715

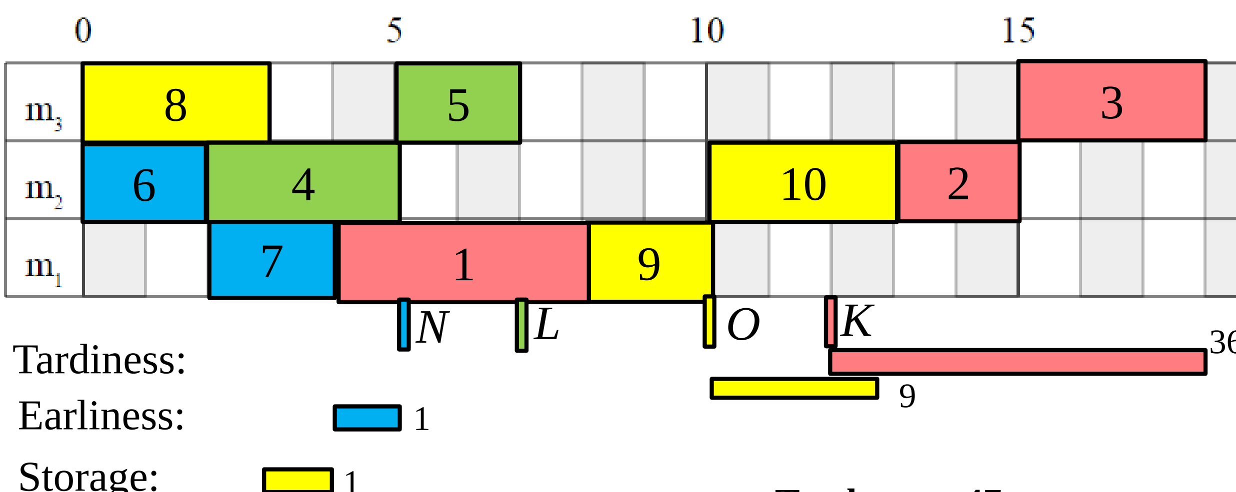
GERAD

Current solution:



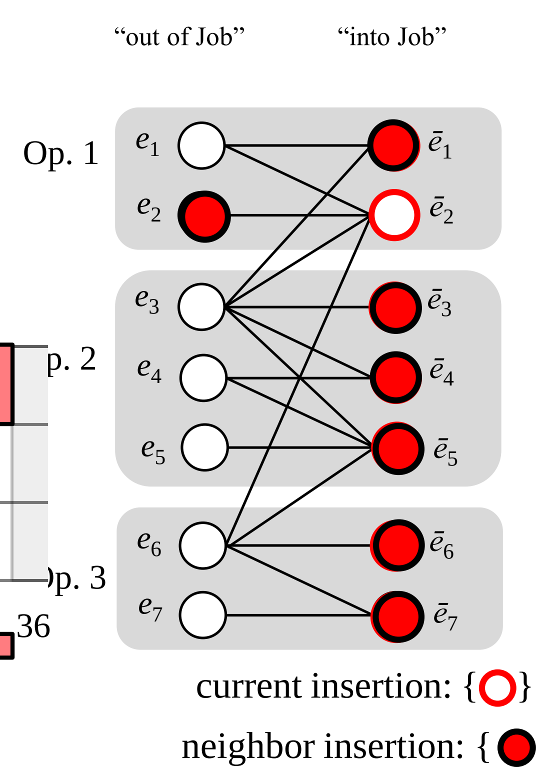
Total costs: 12

Neighbor solution:



Total costs: 47

Conflict Graph H:



Optimal Job Insertion

This problem is NP-hard already in the classical job shop.

- Use conflict graph and its associated MIP formulation to compute an optimal job insertion

Opt. Job Insertion Algorithm

1. Solve MIP
2. *IF* MIP feasible *DO*
3. compute optimal times,
4. and store solution if best,
5. forbid current insertion
6. Go To 1.
7. *ELSE* stop

weights (see later)

$$\text{minimize} \quad \sum_{v \in E^J} c_v x_v$$

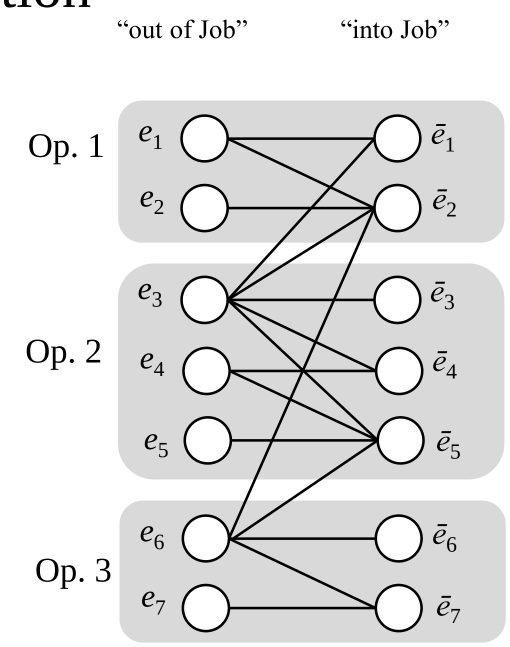
$$\text{subject to:}$$

$$x_w + x_v \le 1 \quad \text{for all } \{v,w\} \in U$$

$$x_v + x_w = 1 \quad \text{for all } \{v,w\} \in \mathcal{E}^J$$

$$x_v \in \{0,1\} \quad \text{for all } v \in E^J$$

Conflict Graph $H = (E^J, U)$:



Opt. Job Insertion Algorithm

1. Solve MIP
2. *IF* MIP feasible *DO*
3. compute optimal times,
4. and store solution if best,
5. forbid current insertion
6. Go To 1.
7. *ELSE* stop

$$\text{minimize} \quad \sum_{v \in E^J} c_v x_v$$

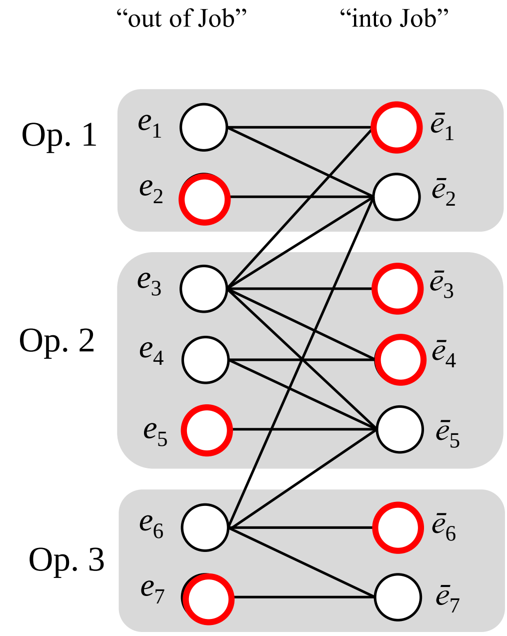
subject to:

$$x_w + x_v \le 1 \quad \text{for all } \{v,w\} \in U$$

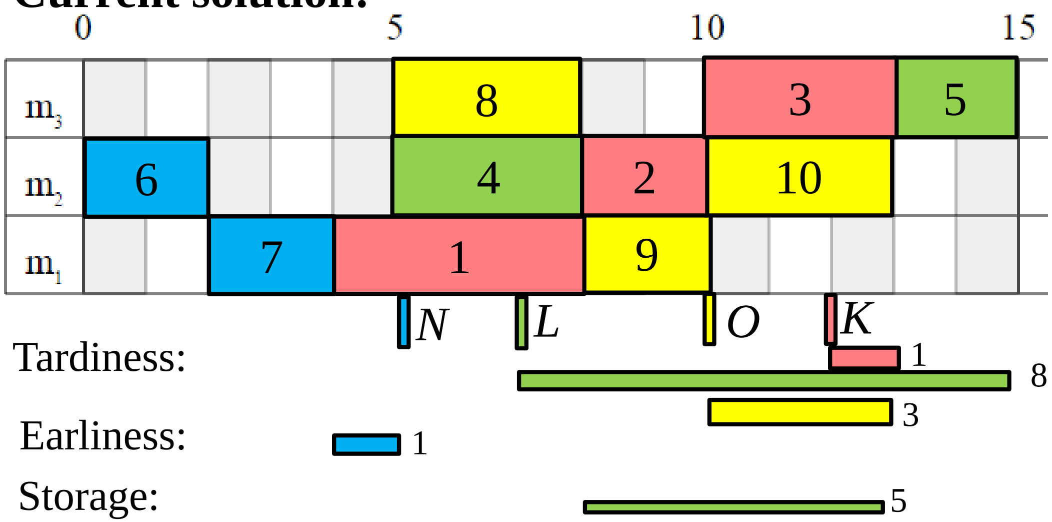
$$x_v + x_w = 1 \quad \text{for all } \{v,w\} \in \mathcal{E}^J$$

$$x_v \in \{0,1\} \quad \text{for all } v \in E^J$$

Conflict Graph $H = (E^J, U)$:



Current solution:



Total costs: 80

Forbid current insertion $I = \{\bigcirc\}$:
(simple way) add constraint:

$$\sum_{v \in I} x_v \le |I| - 1$$

However, then ALL feasible insertions are enumerated! (there are already 25 in the small example)

Opt. Job Insertion Algorithm

1. Solve MIP
2. *IF* MIP feasible *DO*
3. compute optimal times,
4. and store solution if best,
5. forbid current insertion
6. Go To 1.
7. *ELSE* stop

$$\text{minimize} \quad \sum_{v \in E^J} c_v x_v$$

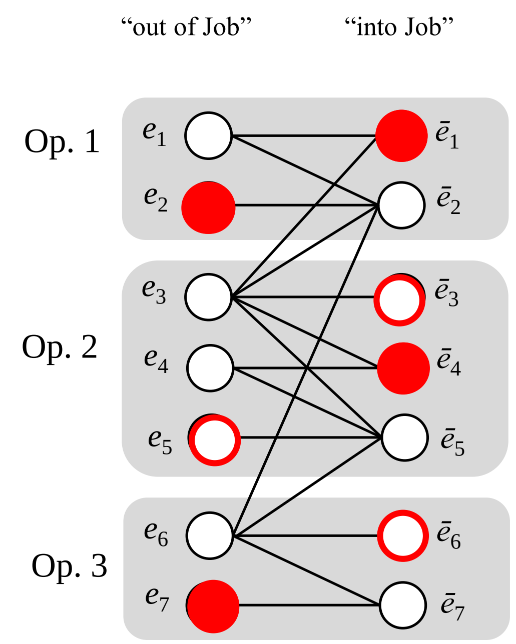
subject to:

$$x_w + x_v \leq 1 \quad \text{for all } \{v,w\} \in U$$

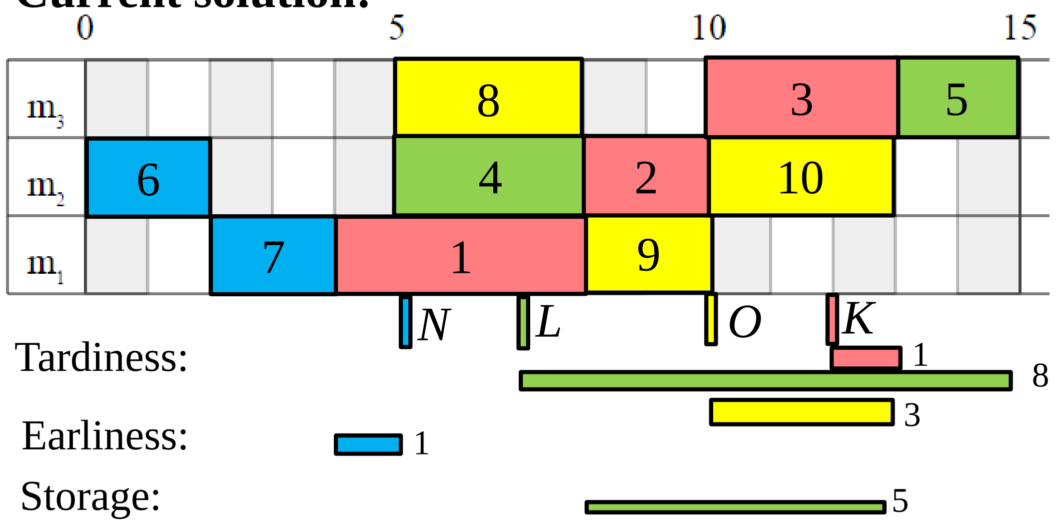
$$x_v + x_w = 1 \quad \text{for all } \{v,w\} \in \mathcal{E}^J$$

$$x_v \in \{0,1\} \quad \text{for all } v \in E^J$$

Conflict Graph $H = (E^J, U)$:



Current solution:



Total costs: 80

Forbid current insertion $I = \{\bigcirc\}$:

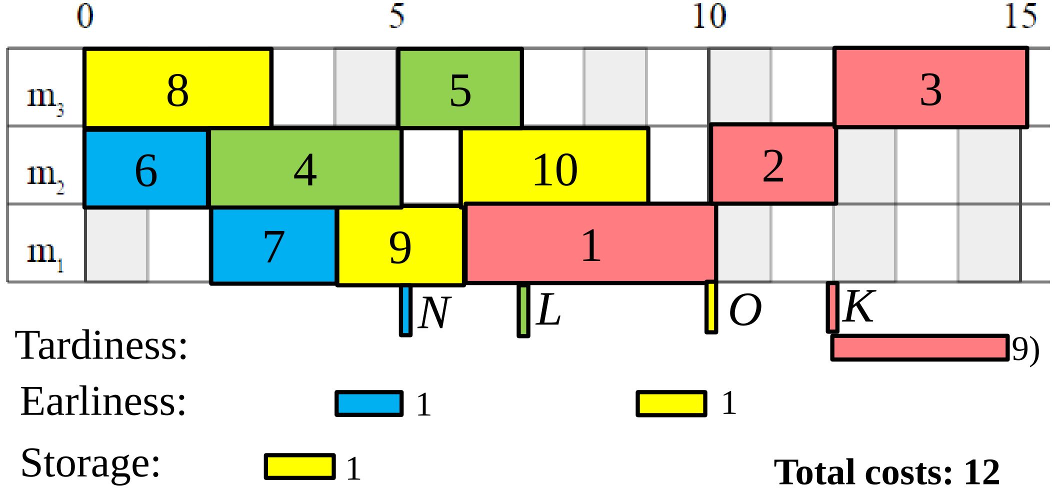
Better: just forbid critical arc set!

$I^{\text{crit}} = \{\bullet\}$: disj. arcs with positive flow in dual solution of timing problem.

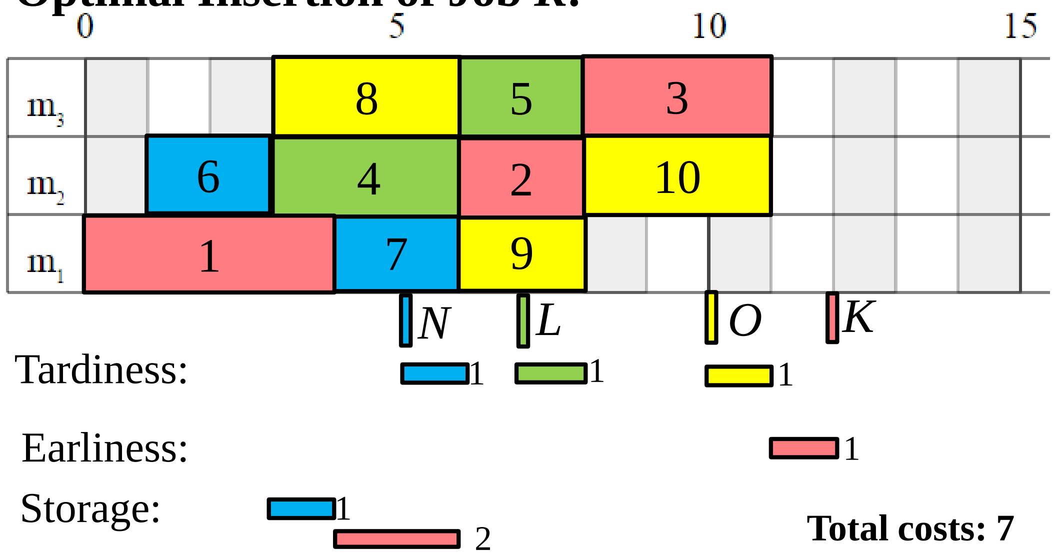
$$\sum_{v \in I^{\text{crit}}} x_v \leq |I^{\text{crit}}| - 1$$

In the example, we generate "just" 12 insertions.

Initial solution:



Optimal Insertion of Job *K*:



Theorem
Opt. Job Insertion Algorithm exactly solves the optimal job insertion problem.

$$\text{minimize} \quad \sum_{v\in E^J} \boxed{c_v x_v}$$

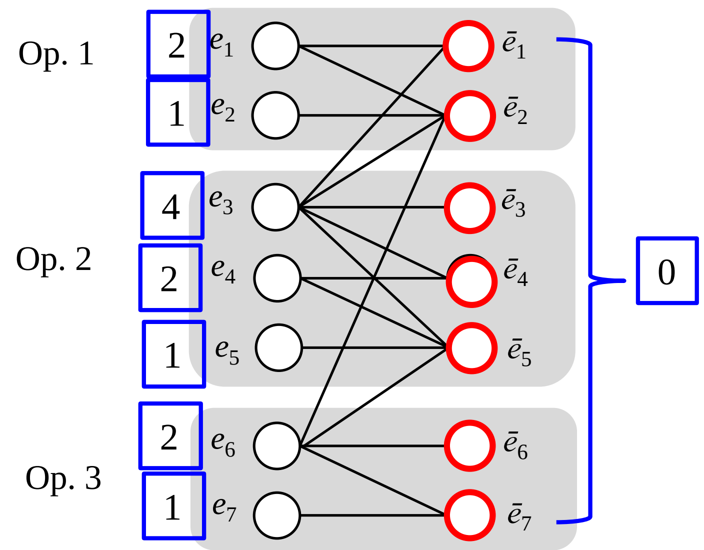
subject to:

$$x_w + x_v \le 1 \quad \text{for all } \{v,w\}\in U$$

$$x_v + x_w = 1 \quad \text{for all } \{v,w\}\in \mathcal{E}^J$$

$$x_v \in \{0,1\} \quad \text{for all } v\in E^J$$

- However, time consuming for medium and large problems
- Locally improving
  - Use weights to generate local neighbors, and stop after a certain time (or number of generated insertions)
  - Obtaining a locally improving neighborhood
- A neighborhood (basic idea): Generate a locally improved neighbor for a subset of jobs and go to best neighbor



4.3. Some Preliminary Computational Results

- Tabu search with swap-based neighborhood
  - A neighbor for each critical arc
- Tabu search with locally improving neighborhood
  - At most 4 jobs are extracted and reinserted
  - At most 150 insertions are considered for each job
- Clearly, tabu search iterations are time consuming

TS iter. / 100 sec.	20 jobs x 5 mach.	20 x 10
reg. objective	50000	3000
swap-based neigh.	140	30
locally improving neigh	4	1.5

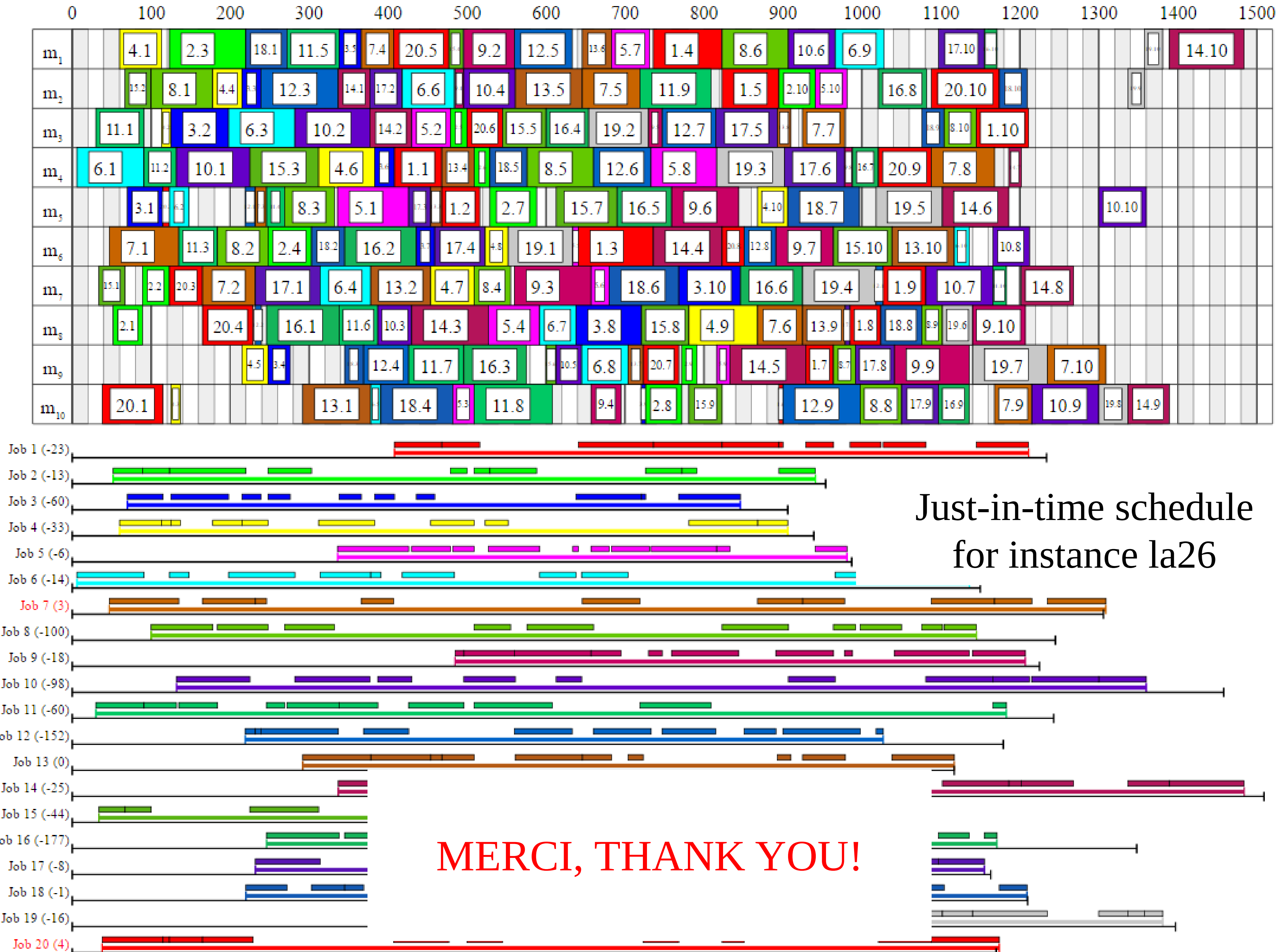
- Hence, very important to
  - Start tabu search with a good initial solution
  - Select moves wisely (and improve implementation!)
- In our example: just-in-time job shop scheduling with squared tardiness costs and linear storage costs
  - Main component are tardiness costs → use solution approach for job shop with regular objectives
  - See, e.g., Bürgy, R. (2016). *A neighborhood for complex job shop scheduling problems with regular objective*. Les Cahiers du GERAD No. G-2016-34. Montreal, Canada
  - Total comp. time 2400 sec. (600 sec. for initial solution computation)

- Comparison with straightforward MIQP
  - MIQP solves some of the smallest instances (la01-la03) to optimality
  - Poor solution quality for larger instances (la26-la40)
- Swap-based neighborhood has a quite good performance
  - (Near-) optimal results in smallest instances
  - Significant improvement of "initial solution" (up to 50%, depending on "tightness" of due dates)
- Locally-improving neighborhood
  - Quite good results in small instances, but "moves too slowly"
  - May be combined with the swap-based neighborhood (adaptive neighborhoods)

Concluding Remarks

- Importance and difficulty of scheduling increases
  - Automated production systems (robots!)
  - Versatile machines, e.g., additive manufacturing (3D printers)
  - Mass customization ("batches of size 1", upward shift of the order penetration point)
  - Smart Manufacturing: pushed by US government (https://www.manufacturing.gov/), Germany (Industry 4.0, http://www.plattform-i40.de/), and others
- We established general models and methods for job shop scheduling
  - First, to the best of our knowledge, considering convex tardiness, earliness, and storage costs
  - More complex process features can be considered as well
- Future work
  - Just-in-time job shop scheduling: Improve details (implementation, parameters, etc.) and use parallelization techniques
  - Apply methods to interesting problems in practice

Just-in-time schedule for instance la26



MERCI, THANK YOU!