ICAPS 2024: Tutorial

AI Techniques for Solving Scheduling Problems

Lucas Kletzander, Nysret Musliu, Florian Mischek

Christian Doppler Laboratory for Artificial Intelligence and Optimization for Planning and Scheduling

Institute of Logic and Computation, DBAI

Faculty of Informatics, TU Wien

Outline

- Scheduling Problems: Case studies
- Solution techniques
  - Solver-independent modelling
  - Constraint programming
  - Metaheuristic techniques
  - Hybrid methods
- Automated algorithm selection and instance space analysis
- Automated algorithm design/Hyper-heuristics
- Industrial applications

Scheduling Problems: Case studies

Investigated Applications in our Lab

Rotating Workforce Scheduling

Shift Design

Break Scheduling

Nurse Rostering

Torpedo Scheduling

Electric Vehicle Charging

Tourist Trip Planning

Social Golfer Problem

High School Timetabling

Production Leveling Problem

Parallel Machine Scheduling

Industrial Oven Scheduling

Physician Scheduling During a Pandemic

Unicost Set Covering

(Hyper)tree Decomposition

Graph Coloring

Traveling Salesman Problem

Vehicle Routing

Sudoku

Bus Driver Scheduling

Test Laboratory Scheduling

Artificial Teeth Production Scheduling

Project Scheduling

Paint Shop Scheduling Problem

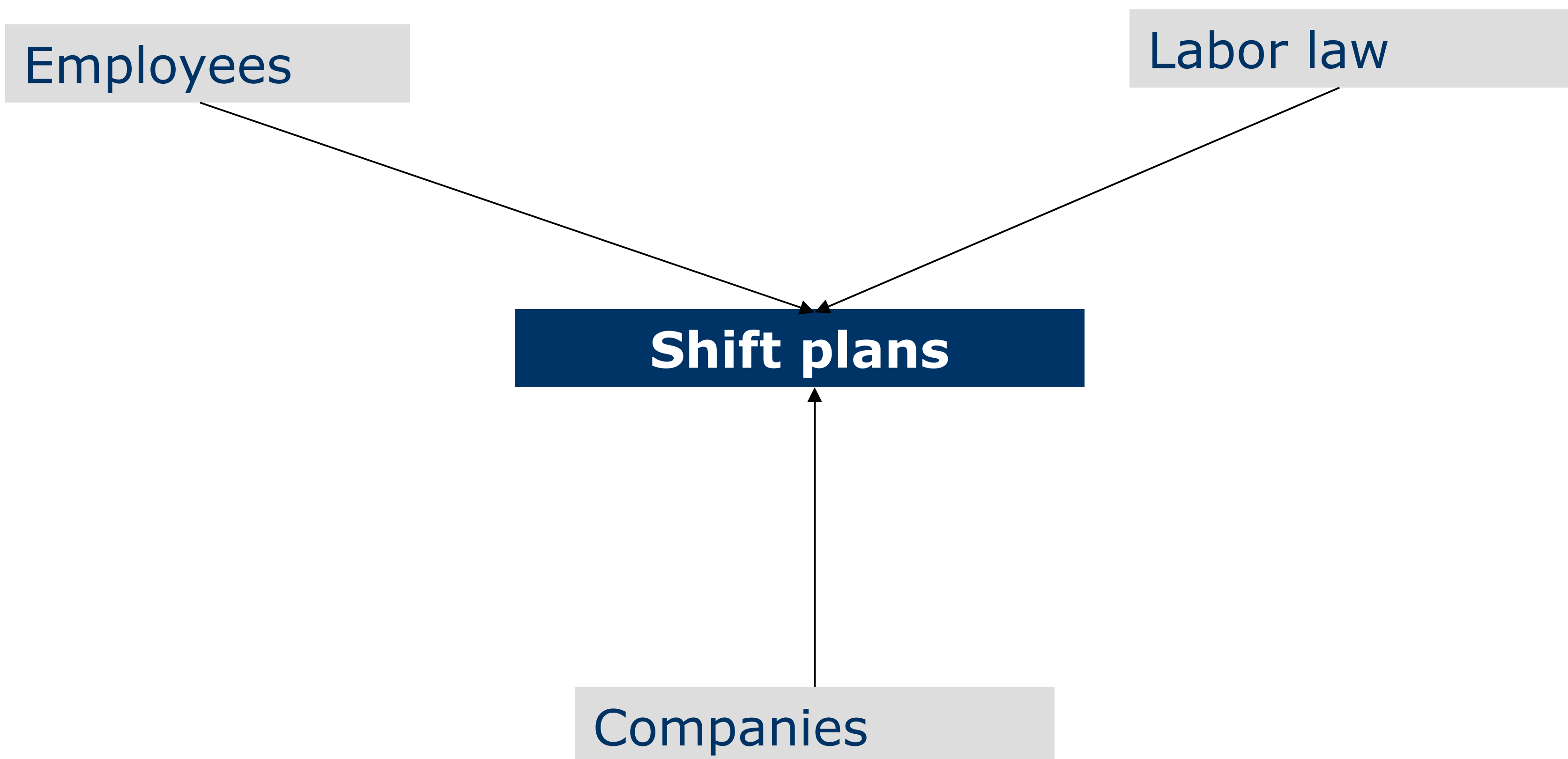
Curriculum-based Course Timetabling

...

Employee Scheduling

- Work schedules influence the lives of employees
- Unsuitable timetable can have a tremendous negative impact on one's health, social life, and motivation at work
- Organizations in the commercial and public sector must meet their workforce requirements and ensure the quality of their services and operations

Employee Scheduling



Employee Scheduling

Real world employee scheduling problems appear in many companies

Airports

Call centers

Air traffic control

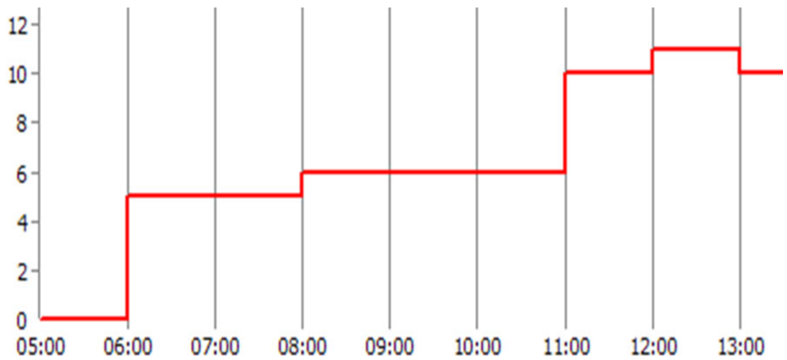
Hospitals

Public transport

Production plants

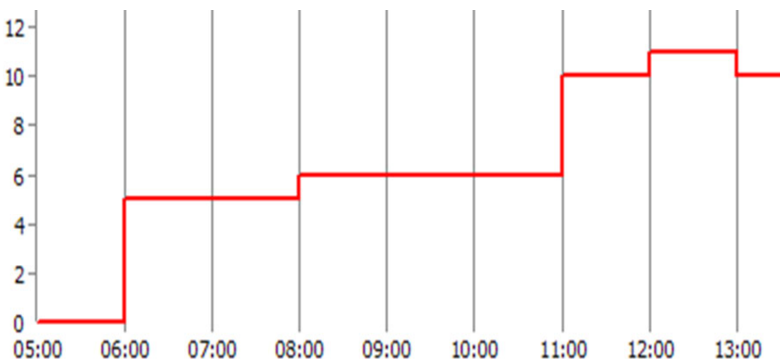
...

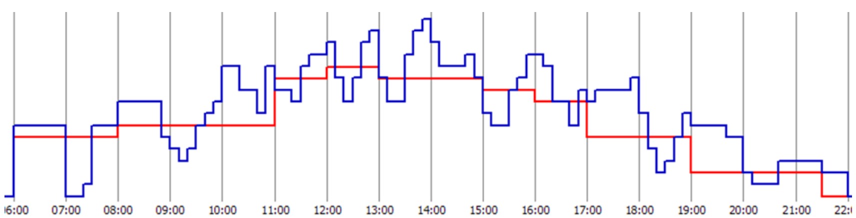
Employee Scheduling Problems

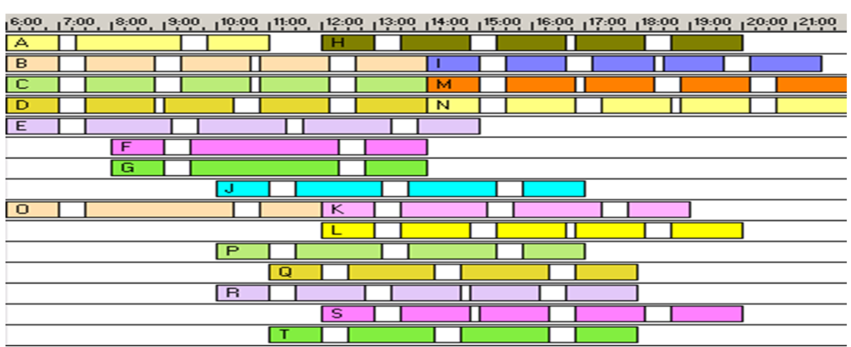


Phase 1:
Workforce requirements

Employee Scheduling Problems



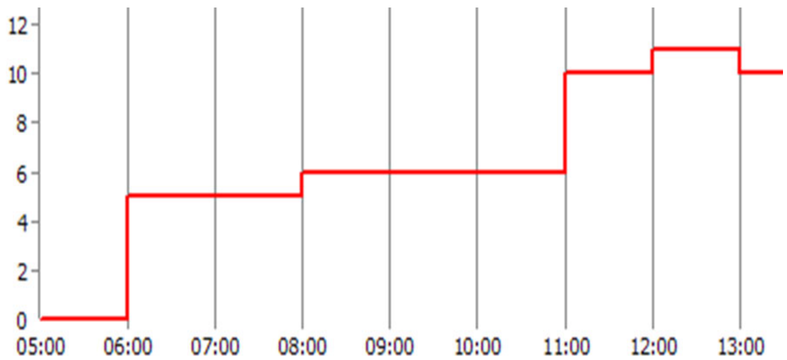


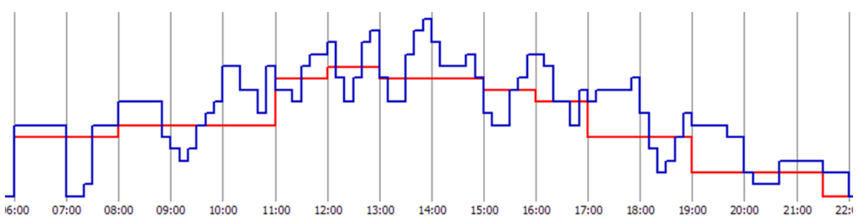


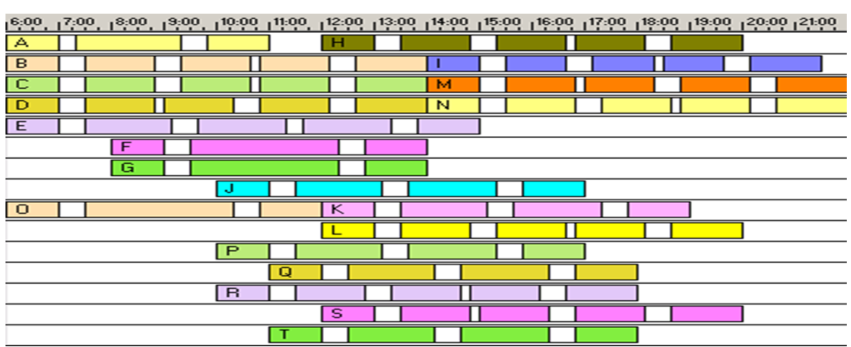
Phase 1:
Workforce requirements

Phase 2:
Shift Design/Break Scheduling

Employee Scheduling Problems







Phase 1:
Workforce requirements

Phase 2:
Shift Design/Break Scheduling

Phase 3:
Assignment of shifts

	Mo	Di	Mi	Do	Fr	Sa	So
A	F	F	F	S	S		
B		N	N	N	N		
C		F	F	N	N	N	N
D			S	S	S	N	N
E	N			F	F	S	S
F	S			F	F	F	F
G	S	S				F	F
H	F	S	S			S	S
I	N	N	N				

Selected papers: [3,4,11,12, 13]

Example: Rotating Workforce Scheduling

Length of schedule: If the schedule is cyclic the total length of a planning period will be: NumberOfEmployees*7

	Mo	Tu	We	Th	Fr	Sa	Su
A	D	D	D			D	D
B	D	D	D	D			
C	A	A	A	A			A
D	A	A	A	A	A		
E	D	D	A	A	A		
F	A	A	N	N	N		
G	N	N	N	N	N		
H		N	N	N	N	N	
I			D	D	D	A	A
J				D	D	N	N
K	N				A	A	N
L	N	N			D	D	D

Number of employees

Employees working shifts:

D: Day shift ; A: Afternoon shift ,

N: Night shift; Day off

Constraints

	Mo	Tu	We	Th	Fr	Sa	Su
A	D	D	D			D	D
B	D	D	D	D			
C	A	A	A	A			A
D	A	A	A	A	A		
E	D	D	A	A	A		
F	A	A	N	N	N		
G	N	N	N	N	N		
H		N	N	N	N	N	
I			D	D	D	A	A
J				D	D	N	N
K	N				A	A	N
L	N	N			D	D	D

Not allowed sequences of shifts:

N - D
N D
A D
N A
N - A
A - D

Maximum and minimum length of periods of successive shifts.

e.g.: N: 2-5, D: 2-6

Temporal requirements: required number of employees in shift *i* during day *j*

Monday (Mo): D: 3, N: 3, A: 3

Maximum and minimum length of work days and days-off blocks

e.g.: days-off block: 2-4

work block: 2-6

Objective

Find a cyclic schedule (assignment of shifts to employees) that satisfies the temporal requirement, and all other constraints

Possible soft constraints:

- Optimization of free weekends (weekends off)
- Optimizing the distribution of weekends
- …

Test Laboratory Scheduling

Input

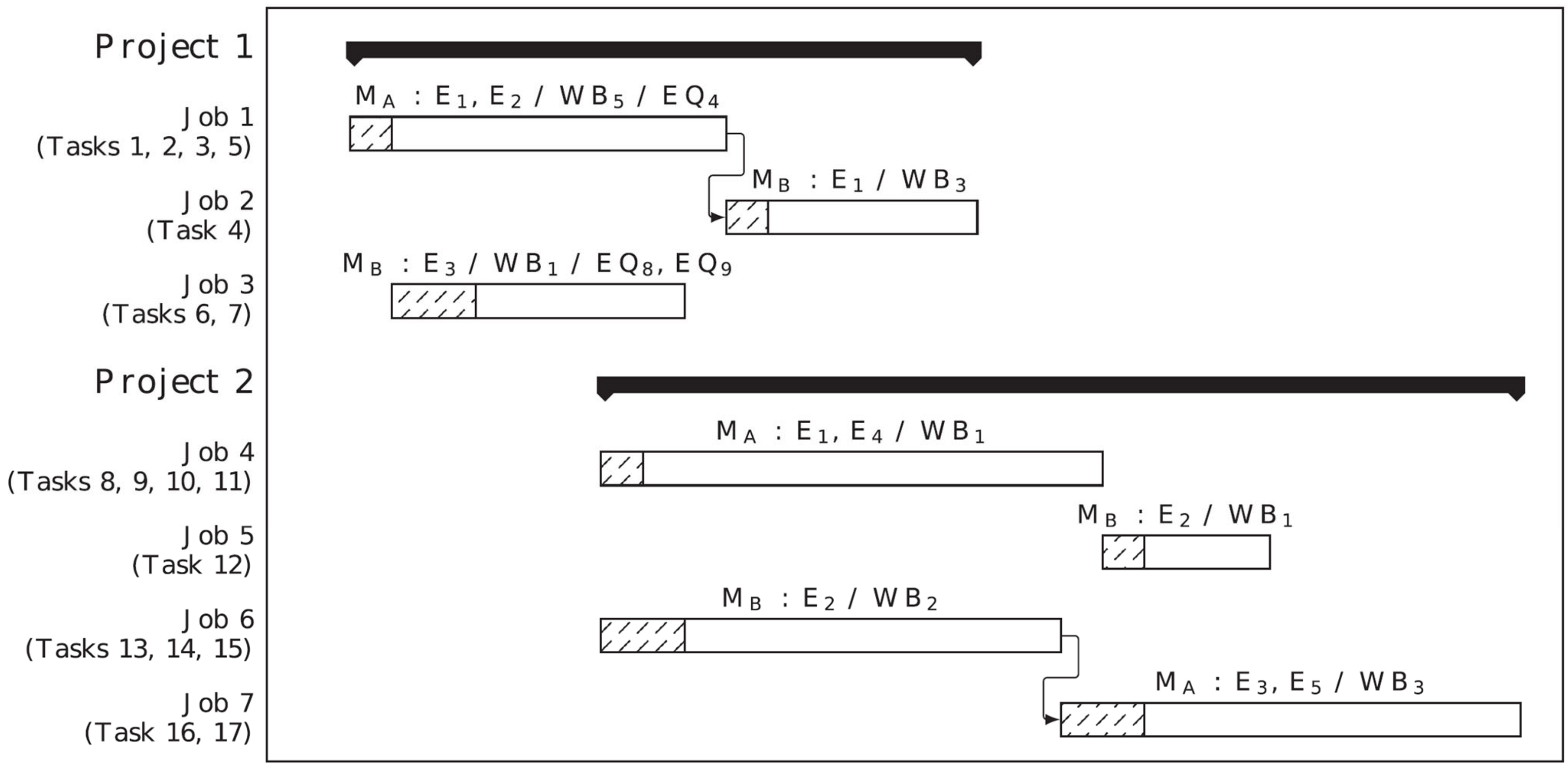
- Scheduling period
- Resources
- Projects and Tasks
- Initial assignments

Solution

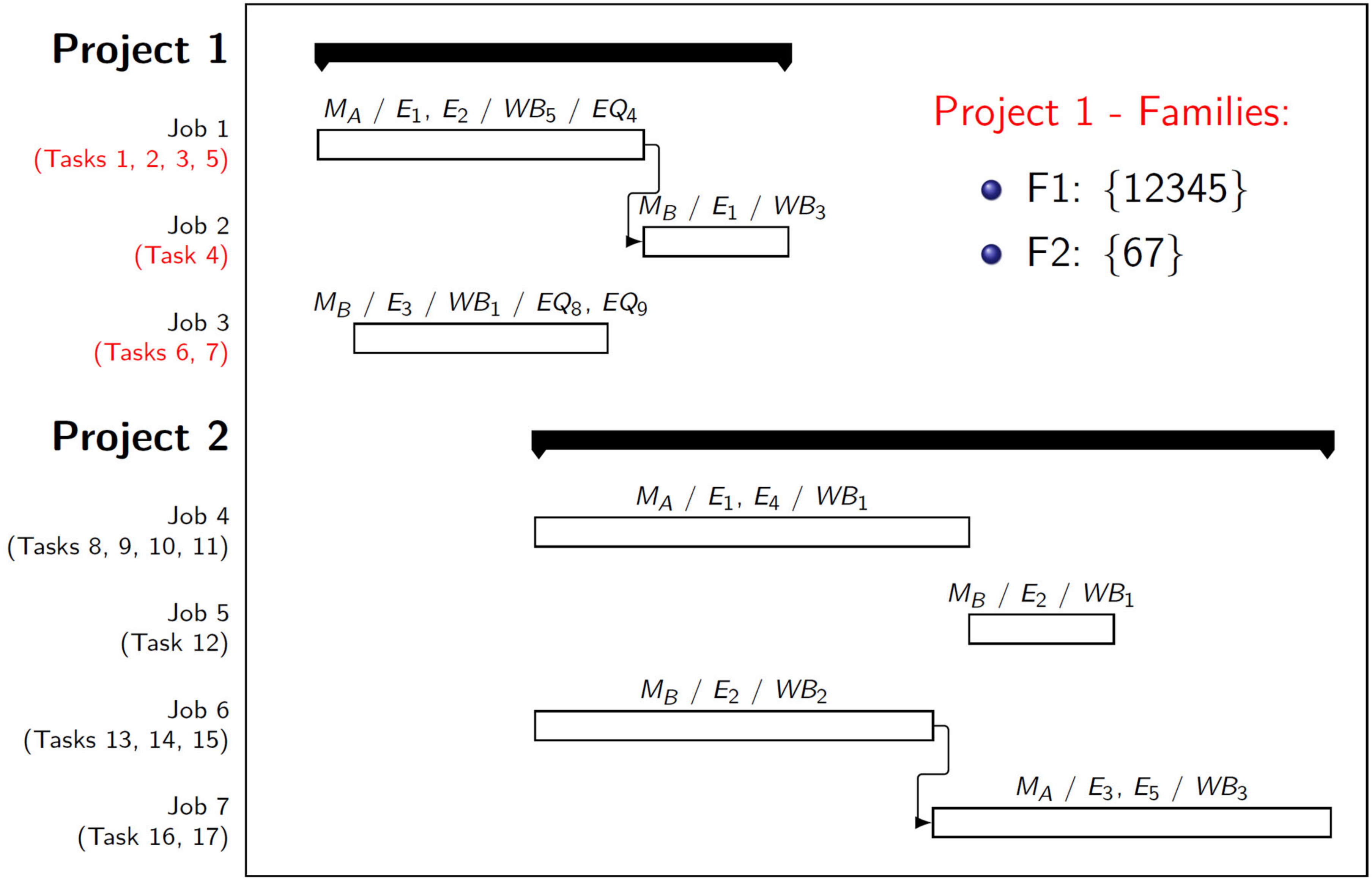
1. *Grouping* of tasks into jobs
2. *Assignment* of
   - Execution mode,
   - Starting timeslot, and
   - Resources

   to each job

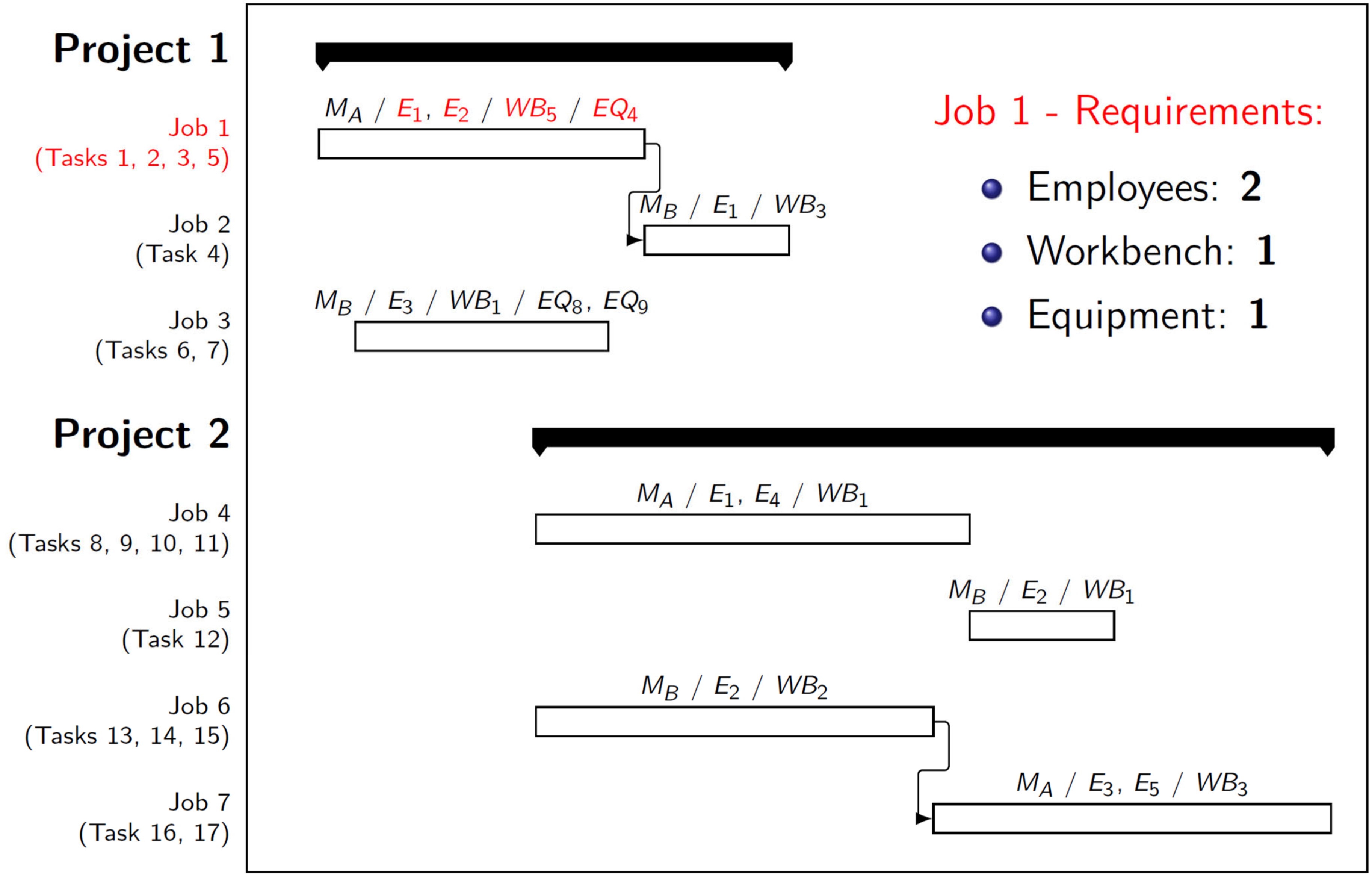
Test Laboratory Scheduling



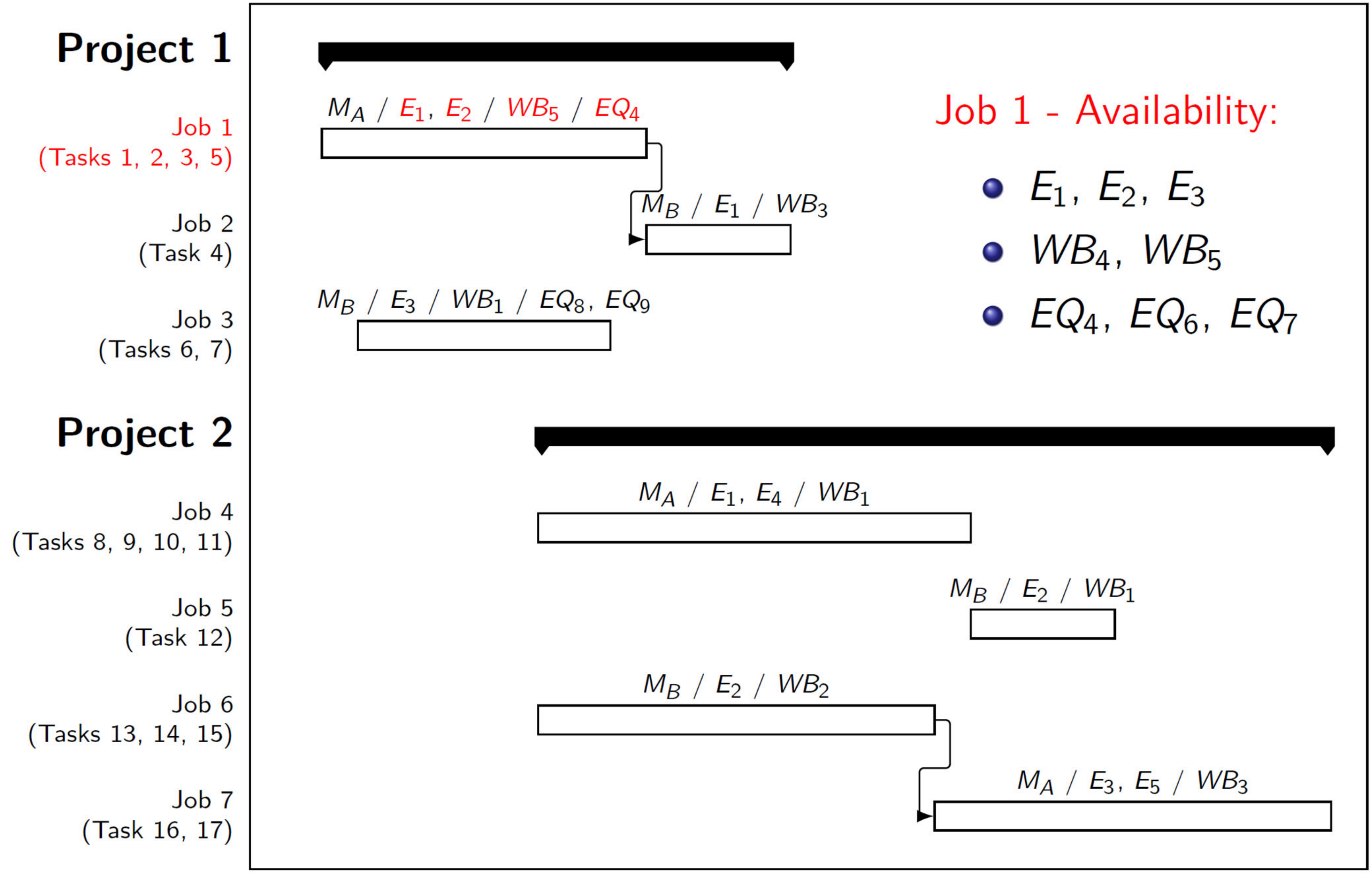
Hard Constraints - Grouping



Hard Constraints - Resource requirements



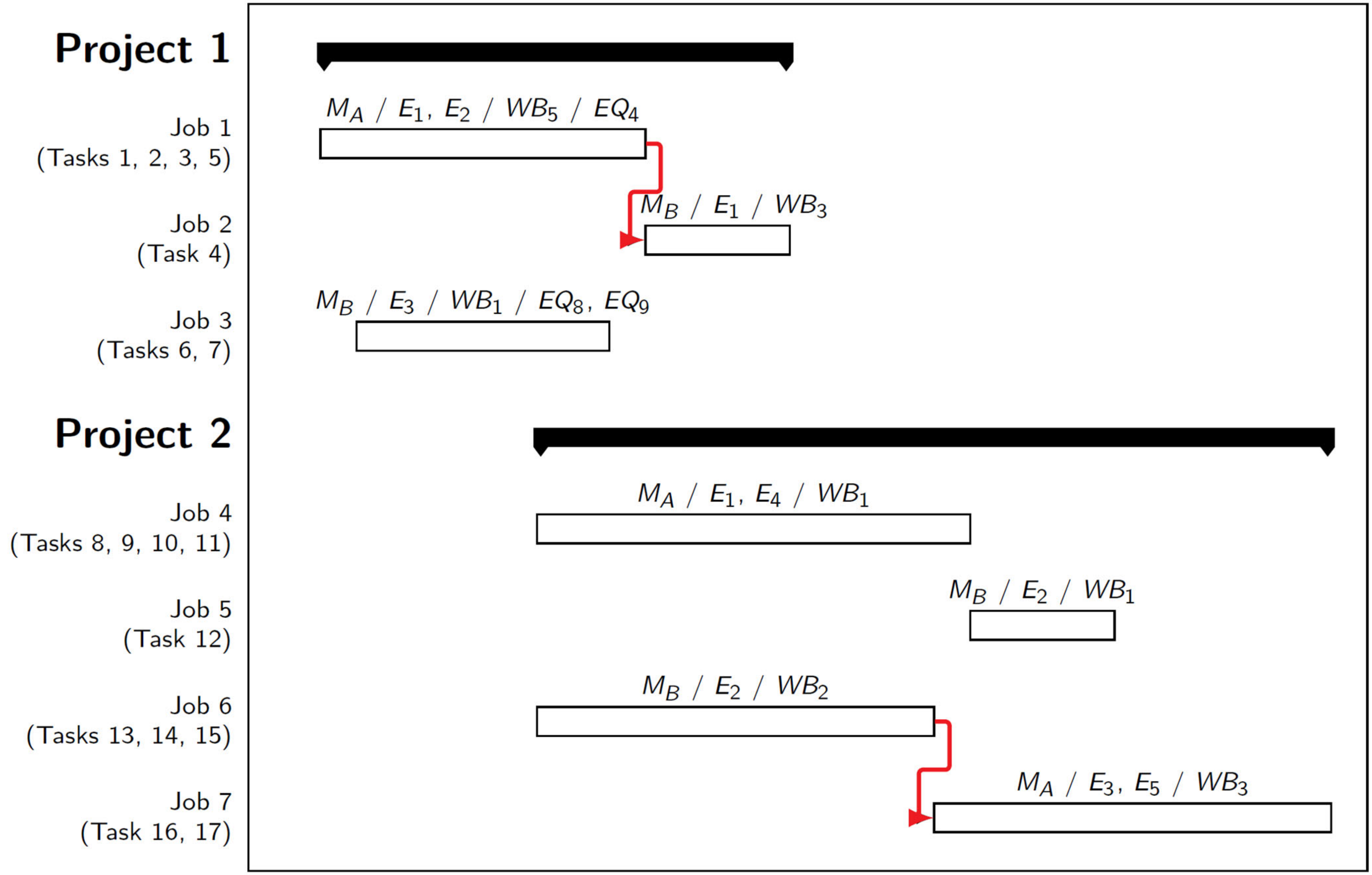
Hard Constraints - Resource availability



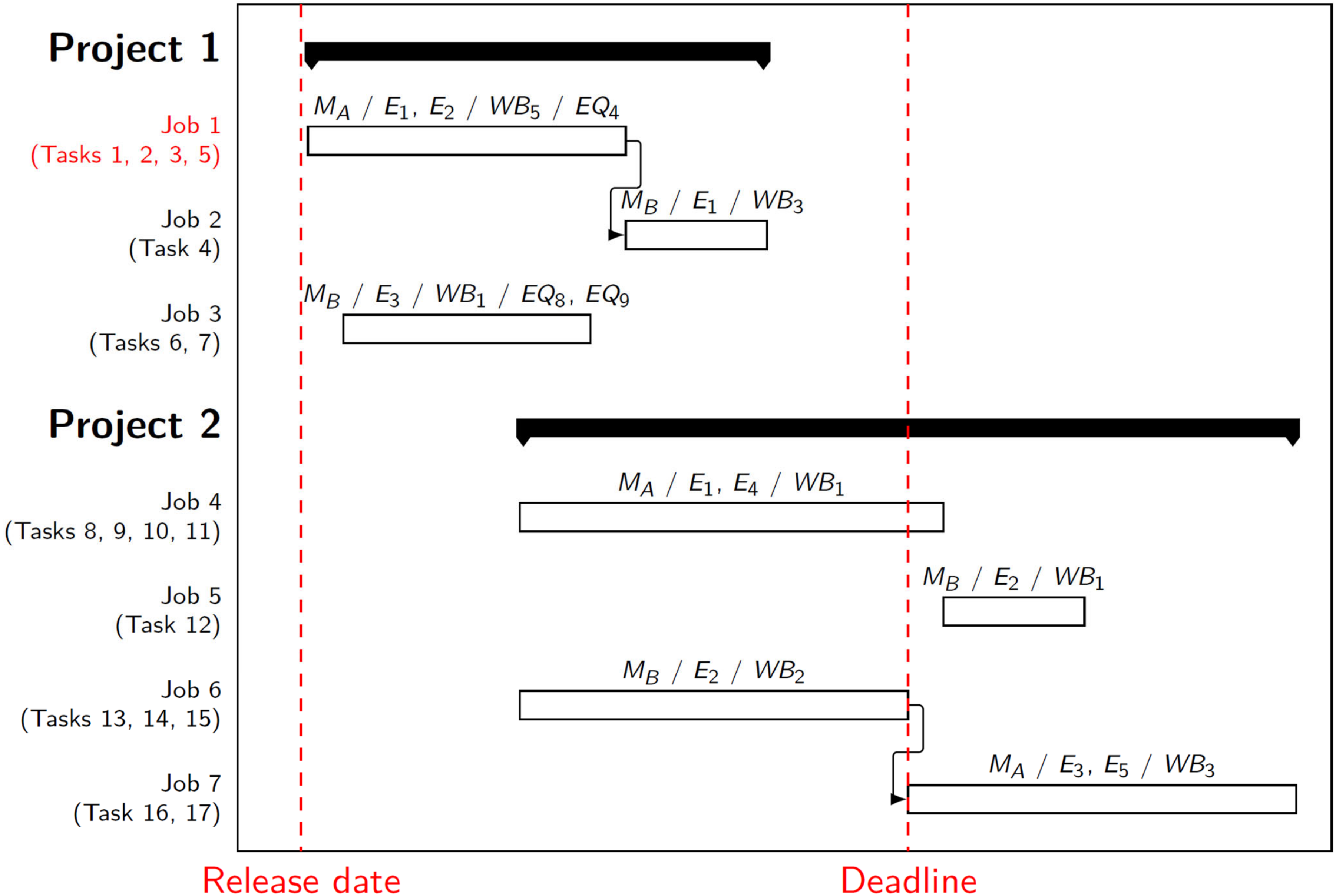
Job 1 - Availability:

- E_1, E_2, E_3
- WB_4, WB_5
- EQ_4, EQ_6, EQ_7

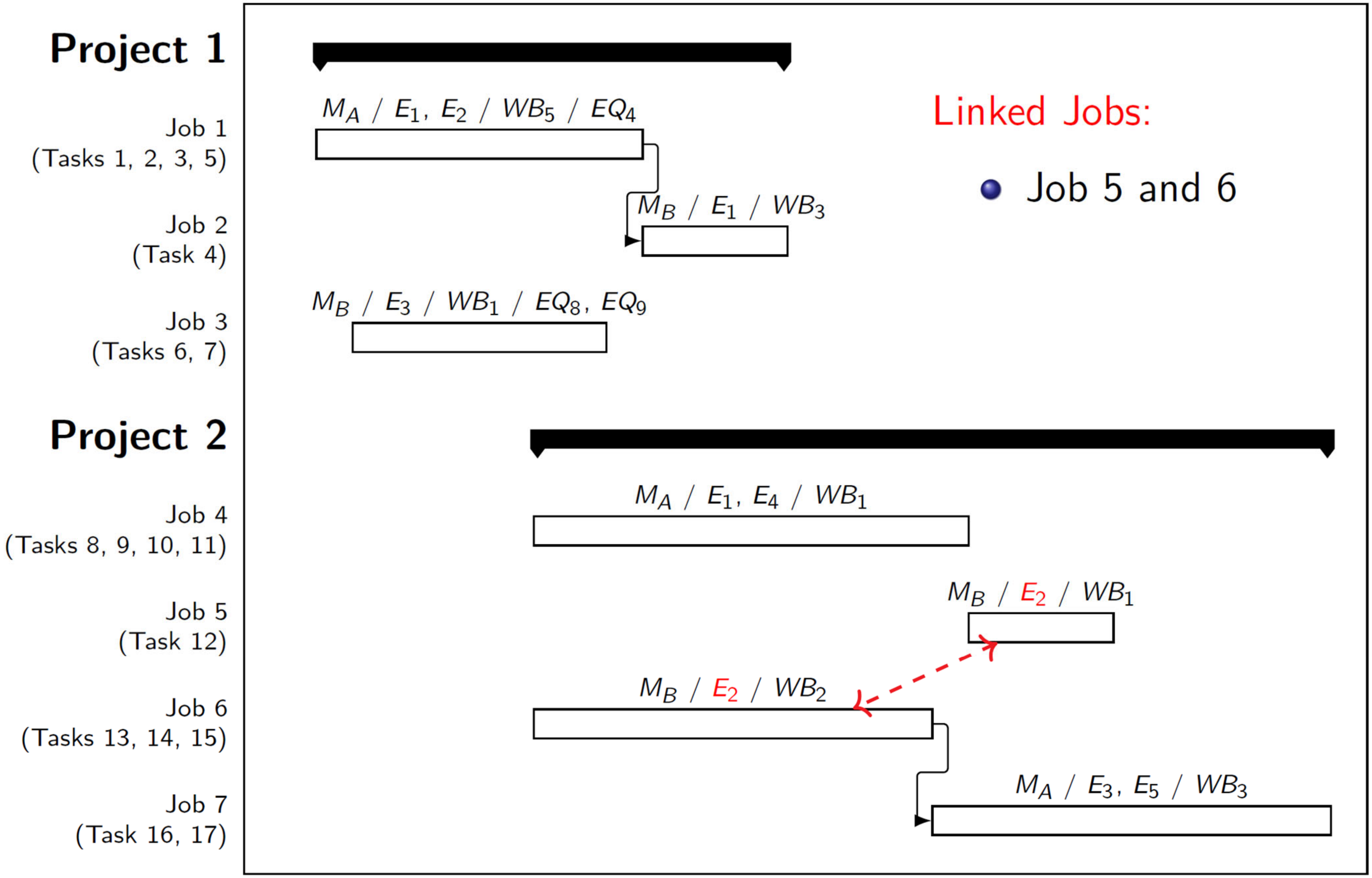
Hard Constraints - Precedence



Hard Constraints - Time Windows



Hard Constraints - Linked Jobs



Linked Jobs:

- Job 5 and 6

Objectives - Number of jobs

Project 1

Job 1
(Tasks 1, 2, 3, 5)

Job 2
(Task 4)

Job 3
(Tasks 6, 7)

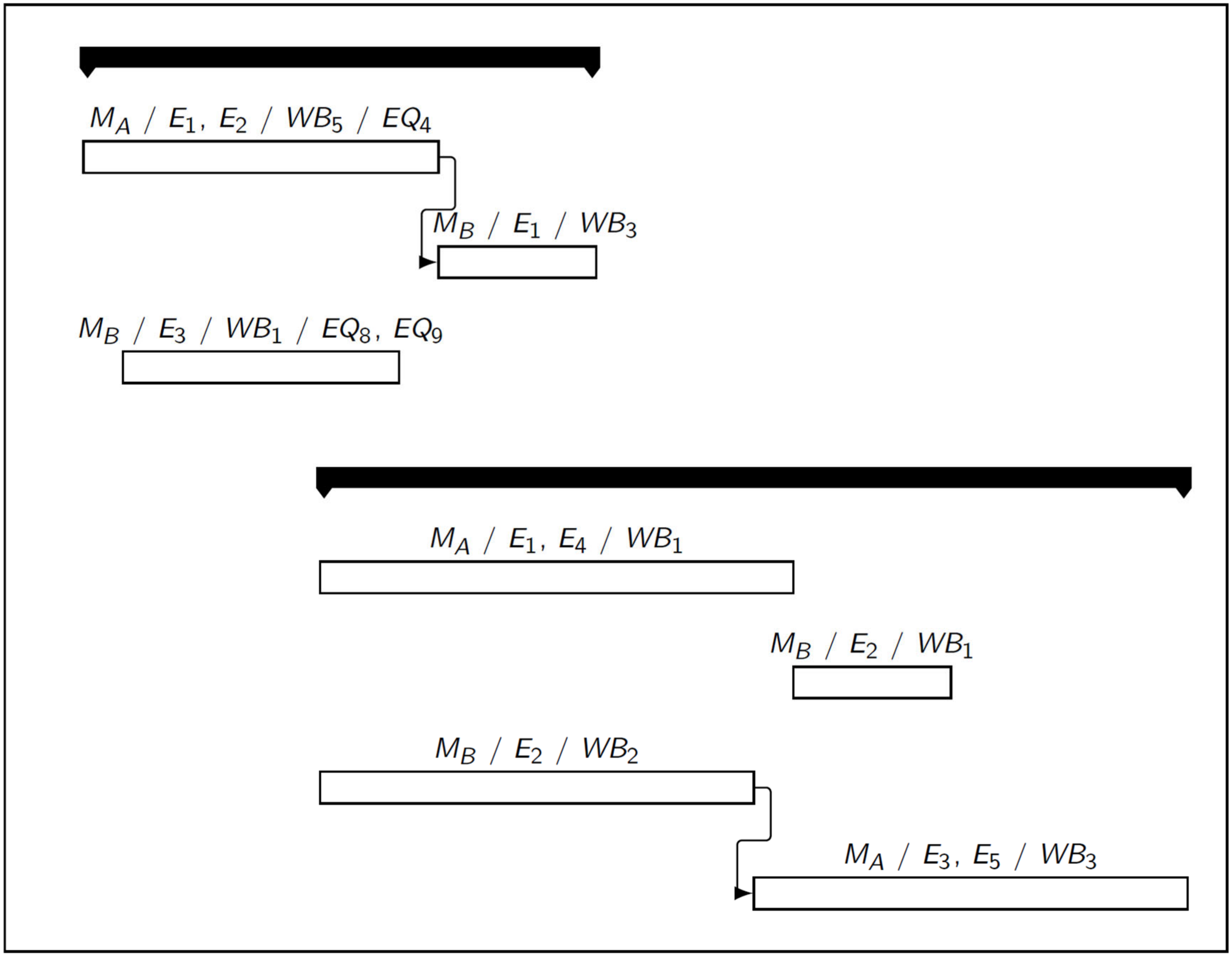
Project 2

Job 4
(Tasks 8, 9, 10, 11)

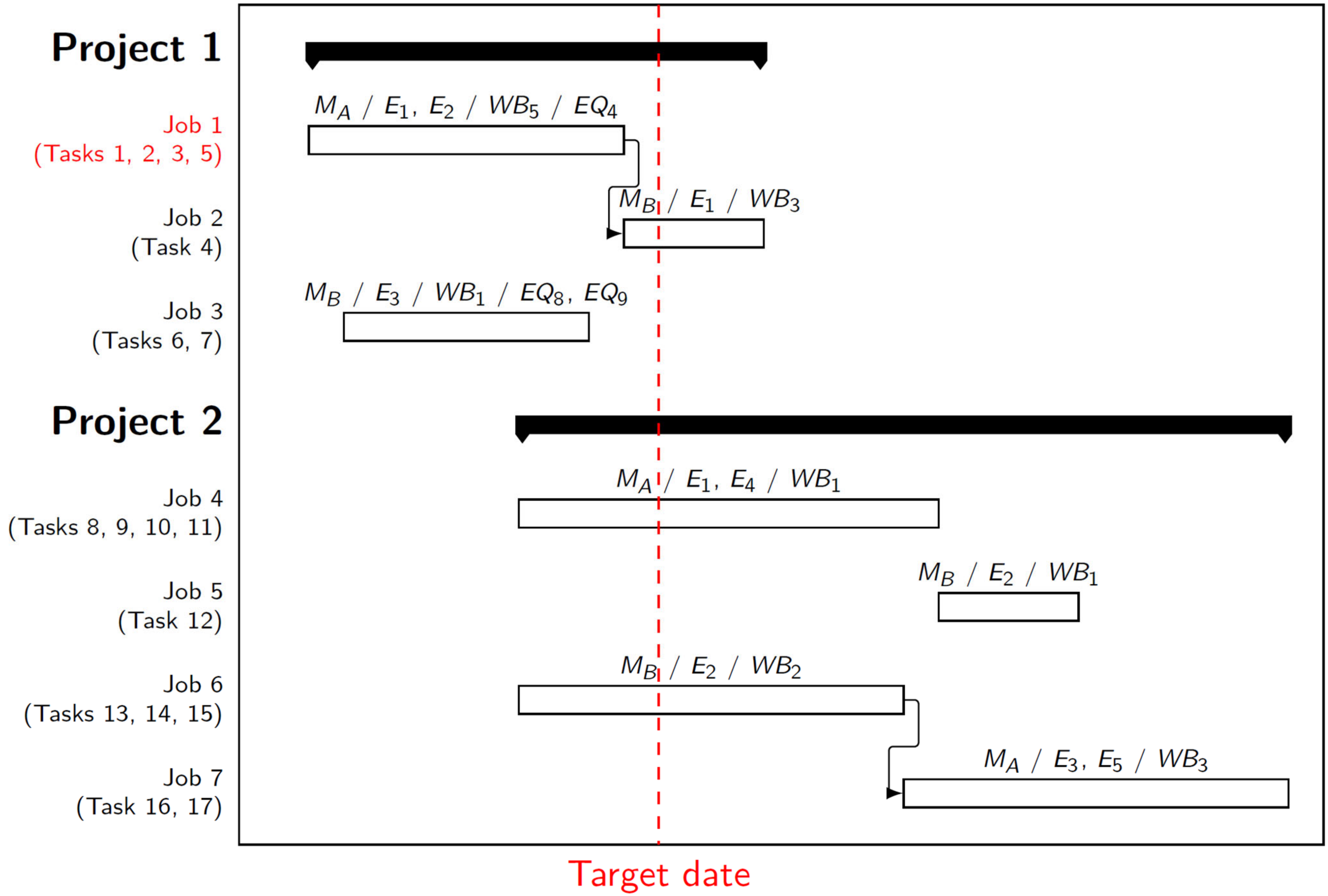
Job 5
(Task 12)

Job 6
(Tasks 13, 14, 15)

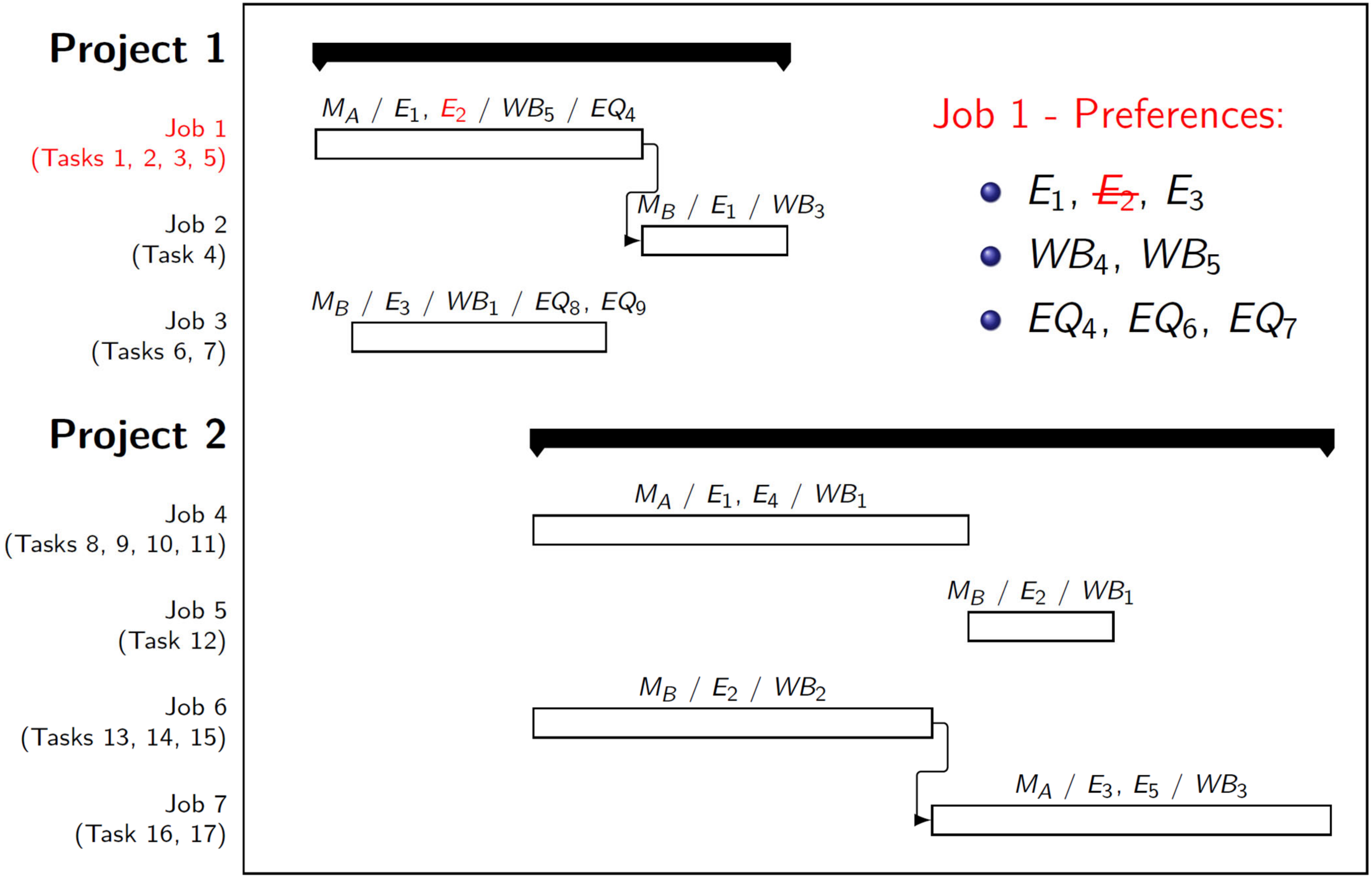
Job 7
(Task 16, 17)



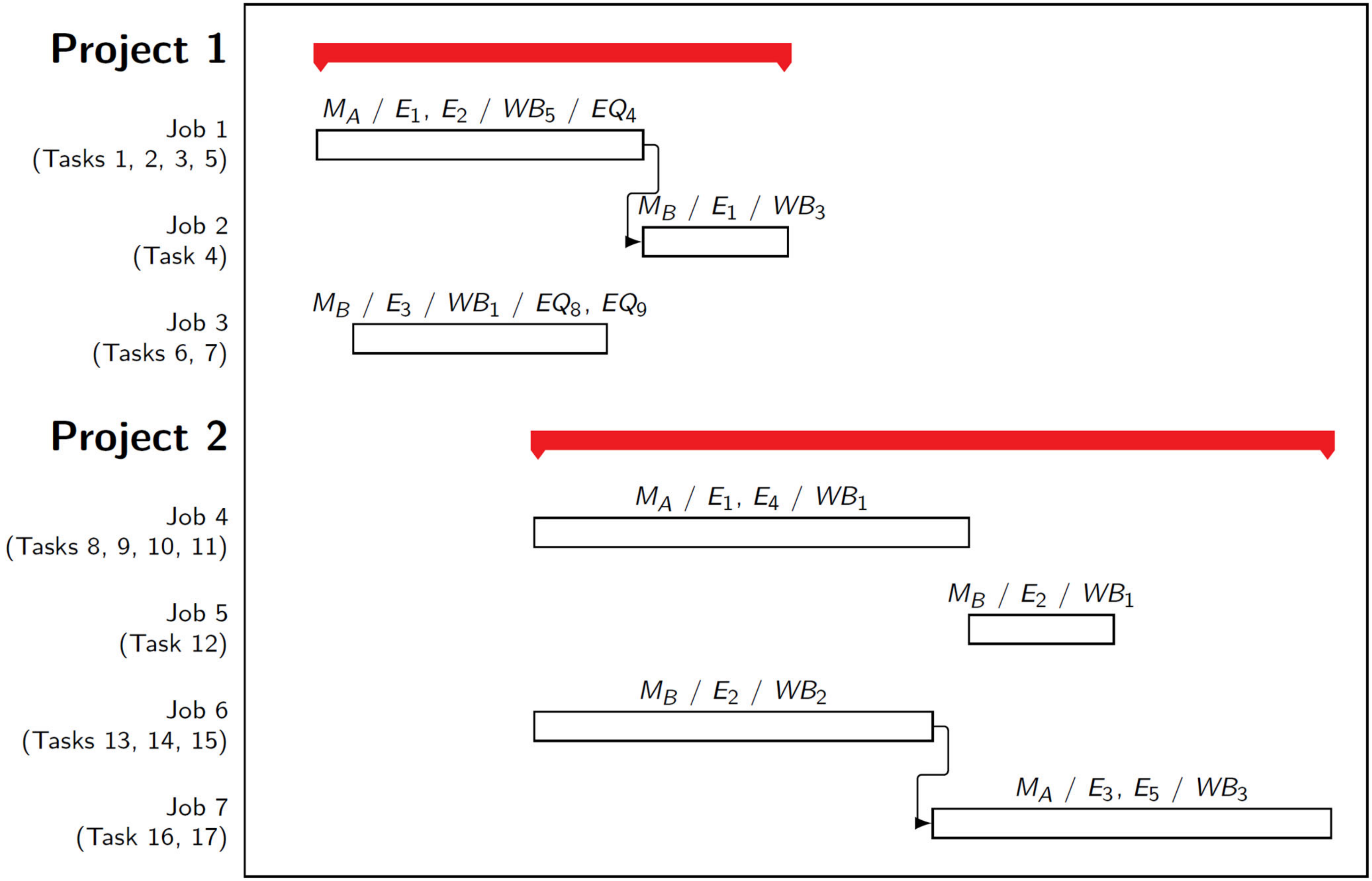
Objectives - Target dates



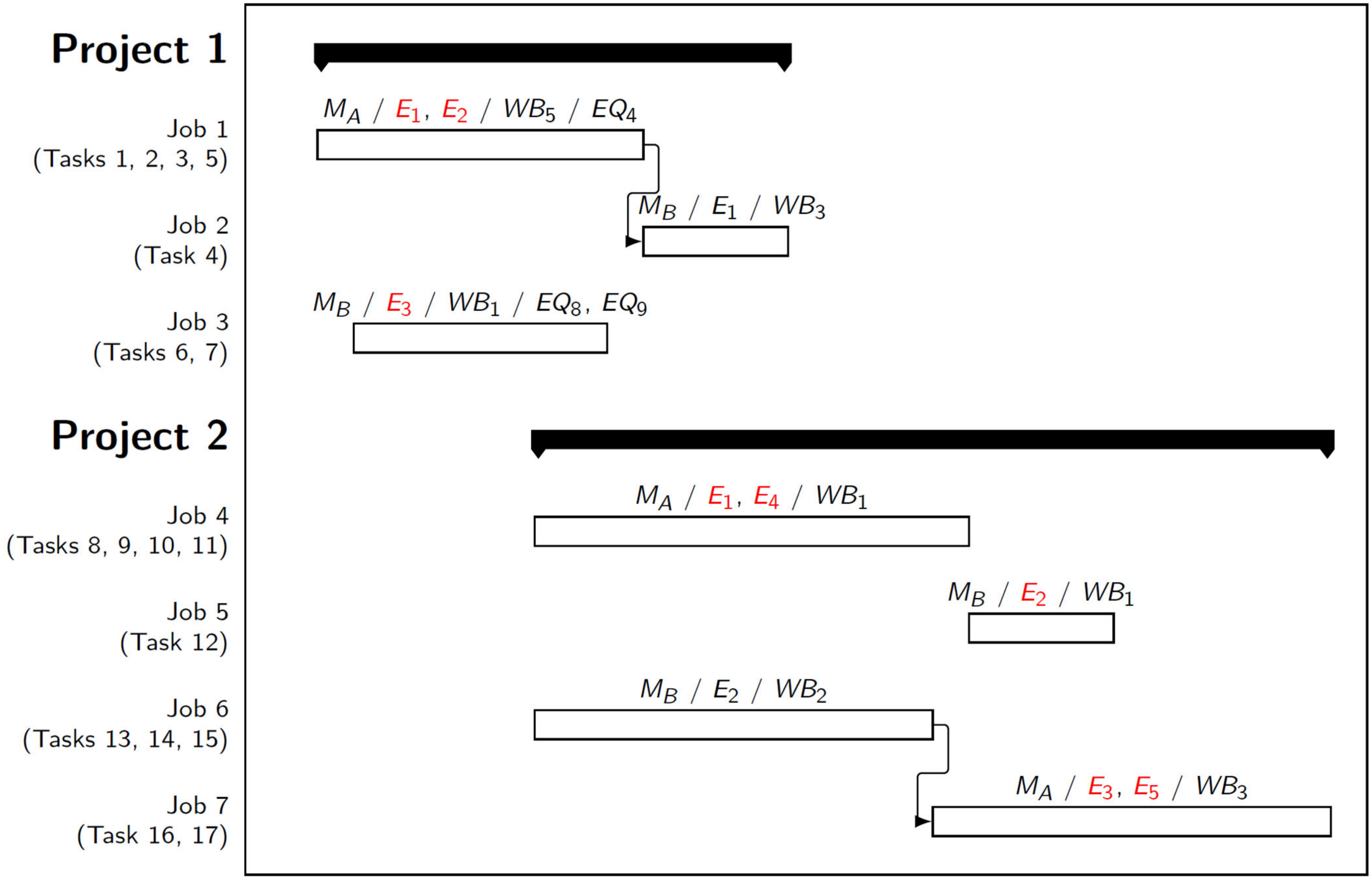
Objectives - Resource preferences



Objectives - Project duration

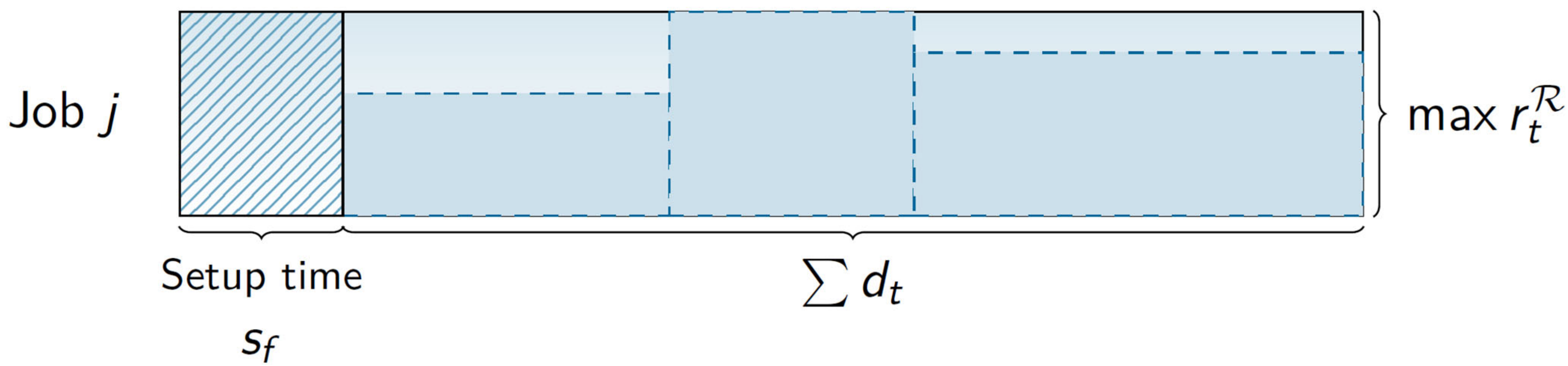


Objectives - Different employees



Task grouping

A job consists of one or several tasks, which define its properties:

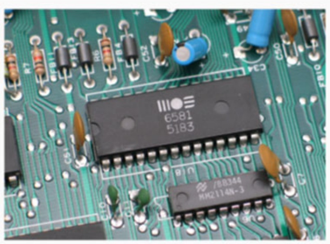


- Available resources: $\mathcal{R}_j = \bigcap \mathcal{R}_t$
- Time window: $\alpha_j = \max \alpha_t$, $\omega_j = \min \omega_t$
- ...

Production Planning and Scheduling

- In these applications it is important to
  - Reduce resource consumption, including energy
  - Increase production efficiency

  …

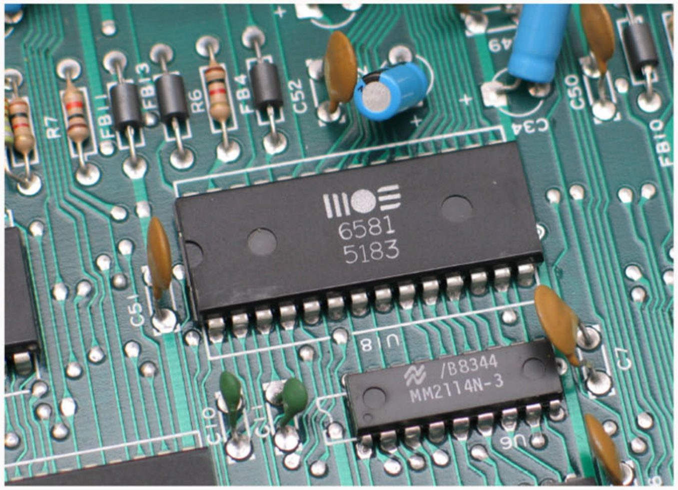


https://commons.wikimedia.org/wiki/File:MOS6581_chtaube061229.jpg, Christian Taube
CC BY-SA 2.5

https://commons.wikimedia.org/wiki/File:Reflow_oven.jpg, Nelatan
CC BY-SA 3.0

	R1	*R2*	*R3*	...
1	A	A	C	...
2	A	A	C	...
3	A	C	C	...
4	B	B	B	...
5	B	B	B	...

Industrial Oven Scheduling



https://commons.wikimedia.org/wiki/File:MOS6581_chtaube061229.jpg, Christian Taube

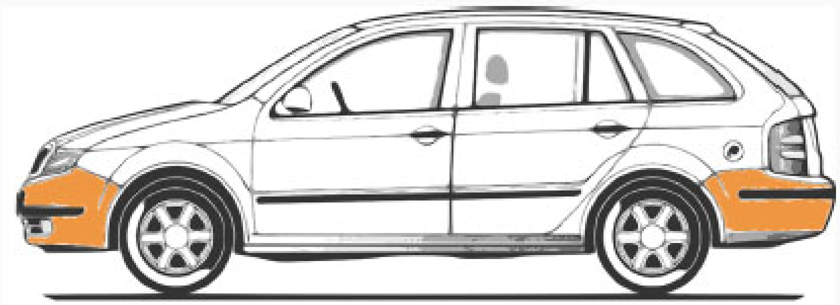
https://commons.wikimedia.org/wiki/File:Reflow_oven.jpg, Nelatan

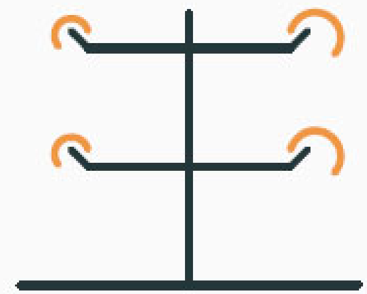
Task: Jobs need to be scheduled and batched efficiently for processing in ovens

Challenge: Many constraints and solution objectives need to be considered

Selected papers: [8]

Paint Shop Scheduling

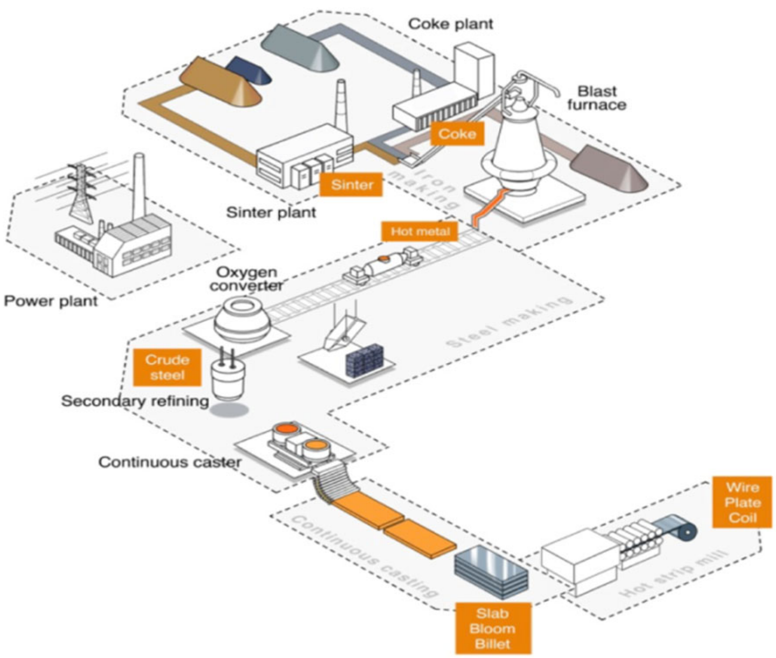




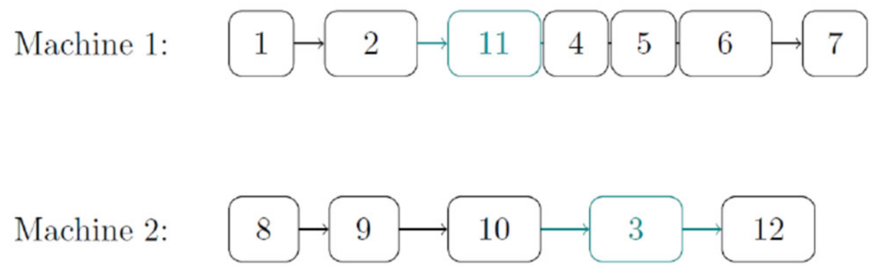
	R1	R2	R3	...
1	A	A	C	...
2	A	A	C	...
3	A	C	C	...
4	B	B	B	...
5	B	B	B	...

Selected papers: [6,7]

Other real-world problems…



Torpedo Scheduling, ACP Challenge, 2016



Parallel Machine Scheduling

Selected papers: [9,10]

…

Other problems…

Time	Monday	Tuesday	Wednesday	Thursday
8:00-9:00	Math	Biology	Math	Math
9:00-10:00	Math	Chemistry	Biology	
10:00-11:00	Physics	Physics		

Week 1			Week 2			Week 3			Week 4					
6	10	12	8	4	6	1	4	2	6	5	14	8	14	13
13	3	4	12	3	7	11	6	15	2	10	7	1	6	3
15	5	1	10	11	5	7	13	9	4	9	11	15	10	9
11	14	7	13	15	2	12	8	5	3	15	8	12	2	11
8	9	2	9	14	1	14	10	3	12	1	13	5	4	7

Selected papers: [24, 25, 26]

https://www.un.org/en/sustainable-development-goals

Solving techniques

AI and optimization methods

Complete approaches

Constraint programming
Answer set programming
SAT/SMT
Mathematical programming
...

Metaheuristic techniques

Tabu search
Simulated annealing
Evolutionary strategies
Memetic algorithms
...

Hybrid methods

Large neighborhood search
Hyper-heuristics
Machine learning based approaches
...

Research work in the CD-Lab Artis

Existing problems

New challenging problems provided by the industry

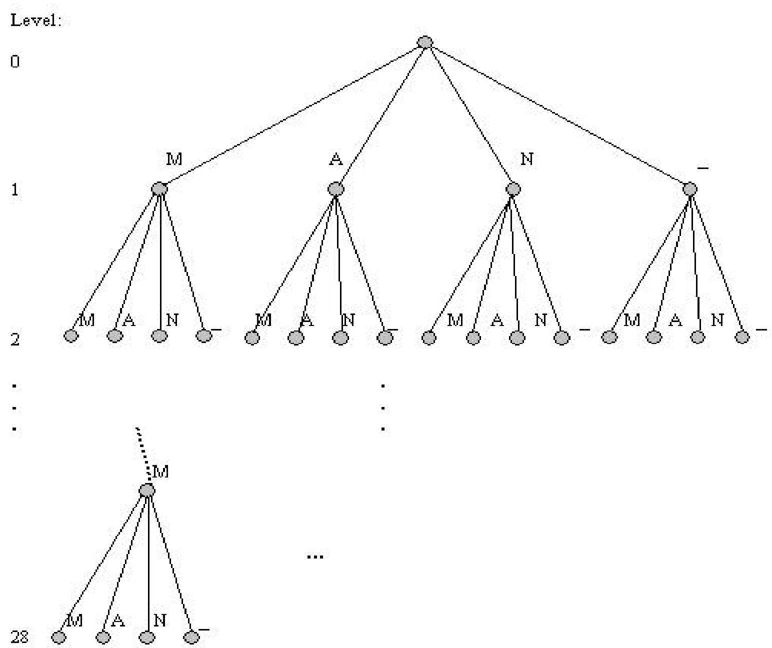
- Formal mathematical formulations
- Identification of related problems in the literature
- Complexity analysis
- General variants of problems
- New problem instances provided to the literature

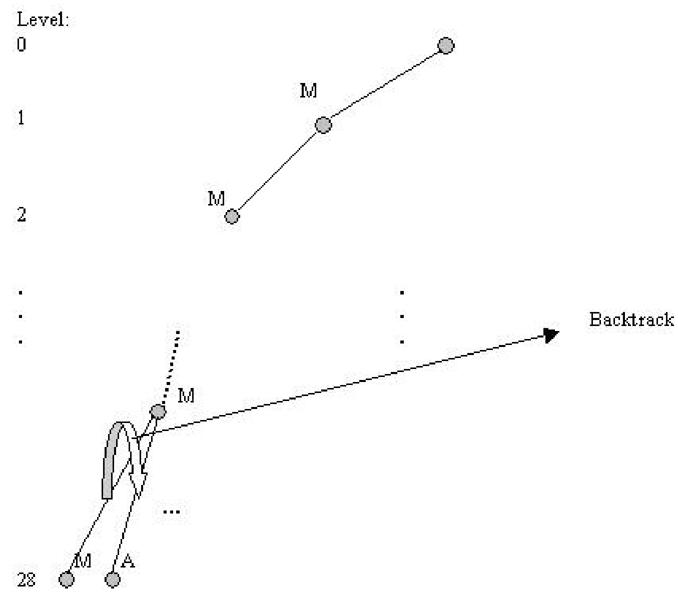
- **Novel modeling**
- **AI solving techniques**
- **Hyper-heuristics**
- **New (hybrid) algorithms**
- **Algorithm selection and instance space analysis**
- **Decision support systems**

https://cdlab-artis.dbai.tuwien.ac.at/

Constraint Programming Techniques

- Tree search
- Constraint propagation
- Forward checking
- Lazy clause generation
- Variable ordering heuristics
- …





Modeling and solvers

- Constraint Programming
  - Solvers: OR-Tools, Chuffed, CP Optimizer…
  - The MiniZinc challenge: https://www.minizinc.org/challenge.html
- Mathematical Programming
  - Solvers: Gurobi, CPLEX…
- Answer Set Programming
  - Solvers: Potassco (the Potsdam Answer Set Solving Collection), DLV, …
- SAT
  - Solvers: http://www.satcompetition.org/
- …

MinZinc

- Constraint modeling language
- Used for modeling constraint satisfaction/optimization problems
  - High-level
  - Solver-independent
    - Model is compiled into FlatZinc that is understood by a wide range of solvers (CP, MIP, …)
- MiniZinc is developed at Monash University
- Free and open-source

Example

Listing 2.1.1: A MiniZinc model aust.mzn for colouring the states and territories in Australia

```
% Colouring Australia using nc colours
int: nc = 3;

var 1..nc: wa;   var 1..nc: nt;  var 1..nc: sa;   var 1..nc: q;
var 1..nc: nsw;  var 1..nc: v;   var 1..nc: t;

constraint wa != nt;
constraint wa != sa;
constraint nt != sa;
constraint nt != q;
constraint sa != q;
constraint sa != nsw;
constraint sa != v;
constraint q != nsw;
constraint nsw != v;
solve satisfy;

output ["wa=\(wa)\t nt=\(nt)\t sa=\(sa)\n",
        "q=\(q)\t nsw=\(nsw)\t v=\(v)\n",
         "t=", show(t),  "\n"];
```

MiniZinc Handbook. Peter J. Stuckey, Kim Marriot, Guido Tack:
https://www.minizinc.org/doc2.2.1/en/MiniZinc%20Handbook.pdf

Rotating Workforce Scheduling: Constraint Programming

- Schedule representation where
  - *w* … the number of days in a week
  - *n* … the number of workers

M	T	W	T	F	S	S	M	T	W	T	F	S	S	…	M	T	W	T	F	S	S
T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}	…	T_{nw-7}	T_{nw-6}	T_{nw-5}	T_{nw-4}	T_{nw-3}	T_{nw-2}	T_{nw-1}

M	T	W	T	F	S	S
$S_{1,1}$	$S_{1,2}$	$S_{1,3}$	$S_{1,4}$	$S_{1,5}$	$S_{1,6}$	$S_{1,7}$
$S_{2,1}$	$S_{2,2}$	$S_{2,3}$	$S_{2,4}$	$S_{2,5}$	$S_{2,6}$	$S_{2,7}$
…						
$S_{n,1}$	$S_{n,2}$	$S_{n,3}$	$S_{n,4}$	$S_{n,5}$	$S_{n,6}$	$S_{n,7}$

$$T_k = S_{1+(k/w),1+(k \mod w)}$$

- Variable domains $\quad T_k \in \{D, A, N, O\}$

N. Musliu, A. Schutt, P. J.Stuckey: Solver Independent Rotating Workforce Scheduling. CPAIOR 2018

Temporal Requirements

- For each day d in the week $\sum_{i \in 1..n} (S_{i,d} = sh) = R_{sh,d}$ where
  - $sh \in \{D, A, N\}$
  - $R_{sh,d}$ … the requirement for shift sh at day d

	M	T	W	T	F	S	S
	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$	$S_{1,4}$	$S_{1,5}$	$S_{1,6}$	$S_{1,7}$
	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$	$S_{2,4}$	$S_{2,5}$	$S_{2,6}$	$S_{2,7}$
				…			
	$S_{n,1}$	$S_{n,2}$	$S_{n,3}$	$S_{n,4}$	$S_{n,5}$	$S_{n,6}$	$S_{n,7}$
D	2	2	2	2	2	2	2
A	1	1	1	1	1	1	1
N	1	1	1	1	1	1	1

$$\sum_{i \in 1..n} (S_{i,7} = D) = 2$$

$$\sum_{i \in 1..n} (S_{i,7} = A) = 1$$

$$\sum_{i \in 1..n} (S_{i,7} = N) = 1$$

Global constraints

- Instead of a set of linear constraints for each day, using one global cardinality constraint

	M	T	W	T	F	S	S
	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$	$S_{1,4}$	$S_{1,5}$	$S_{1,6}$	$S_{1,7}$
	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$	$S_{2,4}$	$S_{2,5}$	$S_{2,6}$	$S_{2,7}$
				…			
	$S_{n,1}$	$S_{n,2}$	$S_{n,3}$	$S_{n,4}$	$S_{n,5}$	$S_{n,6}$	$S_{n,7}$
D	2	2	2	2	2	2	2
A	1	1	1	1	1	1	1
N	1	1	1	1	1	1	1

$$gcc([S_{1,7}, \ldots, S_{n,7}], [D, A, N], [2,1,1])$$

- Redundant constraint

$$gcc([S_{1,d}, \ldots, S_{n,d}], [D, A, N, O], [R_{D,d}, R_{A,d}, R_{N,d}, R_{O,d}])$$

Sequence constraints

- For each day *d* in the schedule
  - Maximal length constraints for *D, A, N,* and *O*
    - For example, 4 for *N*

M	T	W	T	F	S	S	M	T	W	T	F	S	S	…	M	T	W	T	F	S	S
T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}	…	T_{nw-7}	T_{nw-6}	T_{nw-5}	T_{nw-4}	T_{nw-3}	T_{nw-2}	T_{nw-1}

$$\sum_{i \in 0..4} (T_i \neq N) > 0 \qquad \sum_{i \in 0..4} (T_{8+i} \neq N) > 0$$

  - Constraints for the maximal length *maxWB* for work blocks

$$\sum_{i \in 0..maxWB} (T_{d+i} = O) > 0$$

Sequence constraints

- For each day *d* in the schedule
  - Minimal length constraints for *D, A, N,* and *O*
    - For example, 3 for *N*

M	T	W	T	F	S	S	M	T	W	T	F	S	S	…	M	T	W	T	F	S	S
T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}	…	T_{nw-7}	T_{nw-6}	T_{nw-5}	T_{nw-4}	T_{nw-3}	T_{nw-2}	T_{nw-1}

$$T_1 \neq N \wedge T_2 = N \rightarrow \sum_{i \in 1..2} (T_{2+i} \neq N) = 0$$

  - Constraints for the minimal length *minWB* for work blocks

$$T_{d-1} = O \wedge T_d \neq O \rightarrow \sum_{i \in 0..maxWB} (T_{d+i} = O) = 0$$

- For each day *d* in the schedule
  - Forbidden sequences of length 2
    - For example, *ND*

$$T_d = N \rightarrow T_{d+1} \neq D$$

  - Forbidden sequences of length 3
    - For example, *N-D*

$$T_d = N \wedge T_{d+1} = O \rightarrow T_{d+2} \neq D$$

Symmetry Breaking Constraints

- Day-off at the last day in the schedule $R_{O,w} > 0 \rightarrow S_{n,w} = O$
- Work day at the first day in the schedule

$$((\forall sh \in \{D, A, N\}, \forall j \in 2..w : R_{sh,j-1} = R_{sh,j}) \vee (R_{O,1} < R_{O,w})) \rightarrow S_{1,1} \neq O$$

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
Fred	$\boxed{\text{not } O}$	?	?	?	?	?	?
Alex	?	?	?	?	?	?	?
John	?	?	?	?	?	?	?
Emma	?	?	?	?	?	?	?
Mark	?	?	?	?	?	?	?
Mia	?	?	?	?	?	?	$\boxed{O}$
D	2	2	2	2	2	2	2
A	1	1	1	1	1	1	1
N	1	1	1	1	1	1	1

MiniZinc model

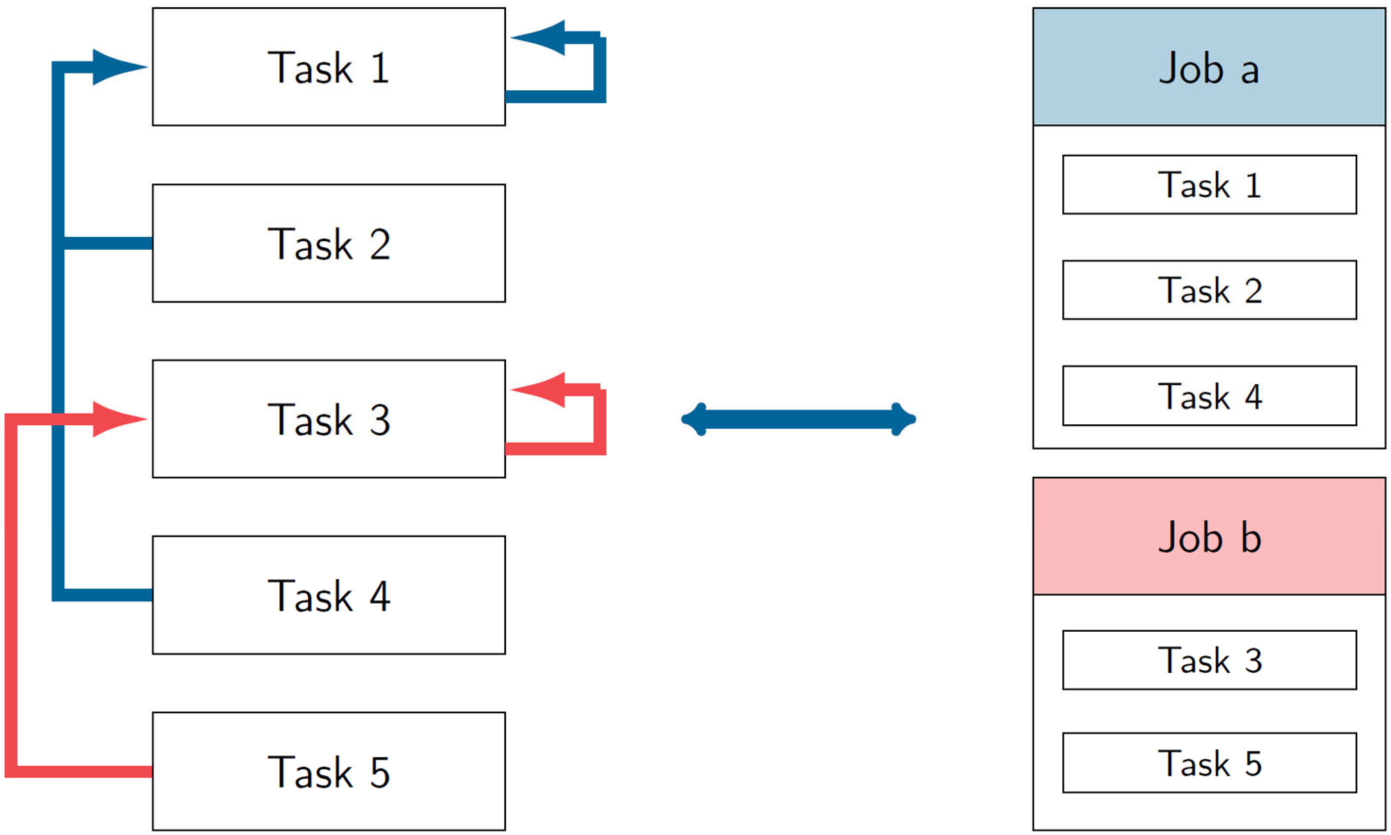
https://www.minizinc.org/challenge2018/results2018.html

Download all problems -> rotating-workforce

Test Laboratory Scheduling: Constraint Programming

Major challenge: Representing grouping

Solution: Representative task for each job



P. Danzinger, T. Geibinger, D. Janneau, F. Mischek, N. Musliu, C. Poschalko: A System for Automated Industrial Test Laboratory Scheduling. ACM Trans. Intell. Syst. Technol. (2023)

Example Constraints

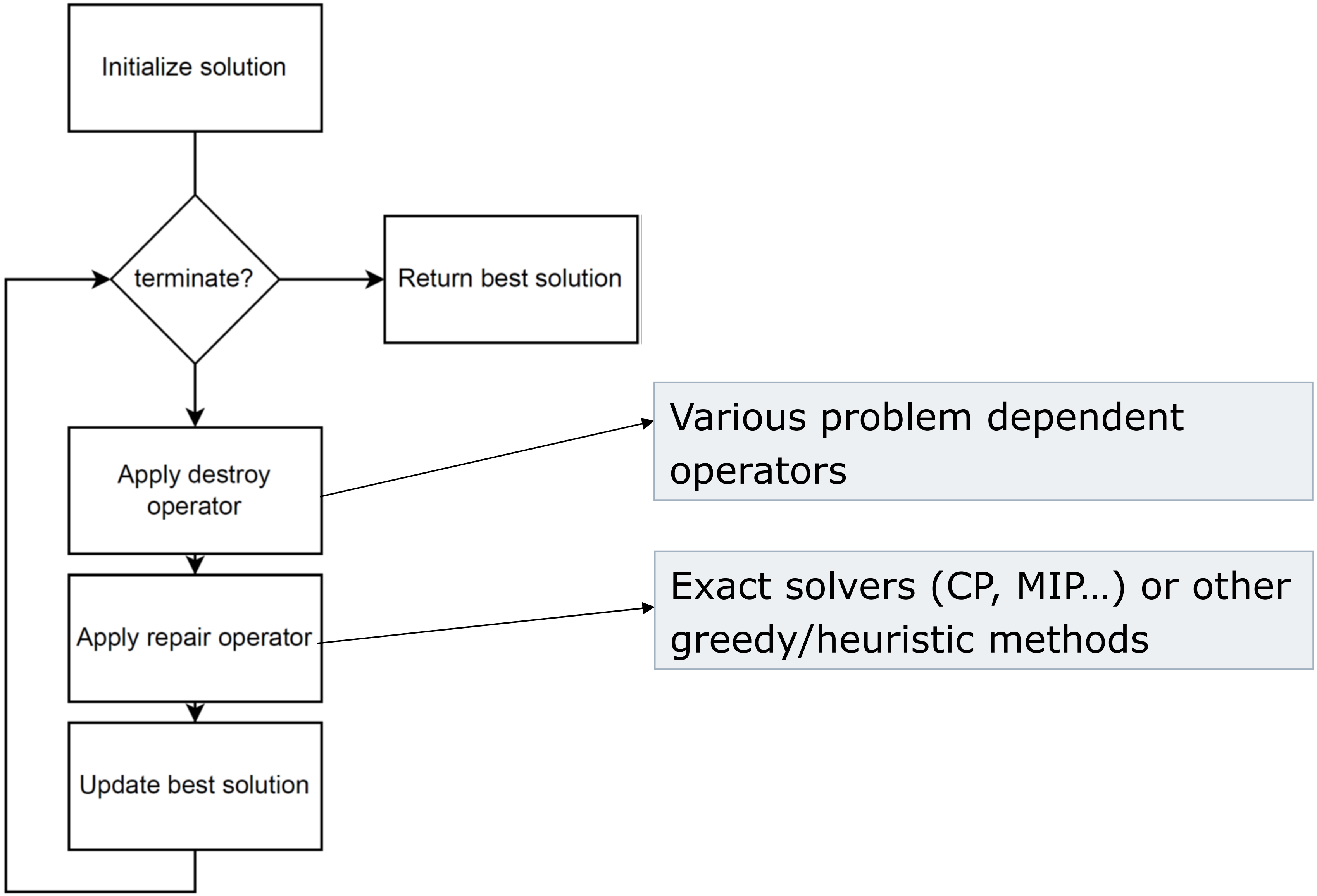
Resource availability:

$$\text{assigned}[\text{repr}[t], r] = 1 \implies r \in \mathcal{R}_t$$
$$\forall t \in \text{Tasks}, r \in \text{Resources}$$

Resource requirements:

$$\sum_{r \in \text{Resources}} \text{assigned}[t, r] = \begin{cases} \max_{t' \in \text{Tasks}:\text{repr}[t'] = t} |Req_{t'}| & \text{if } \text{repr}[t] = t \\ 0 & \text{otherwise} \end{cases}$$
$$\forall t \in \text{Tasks}$$

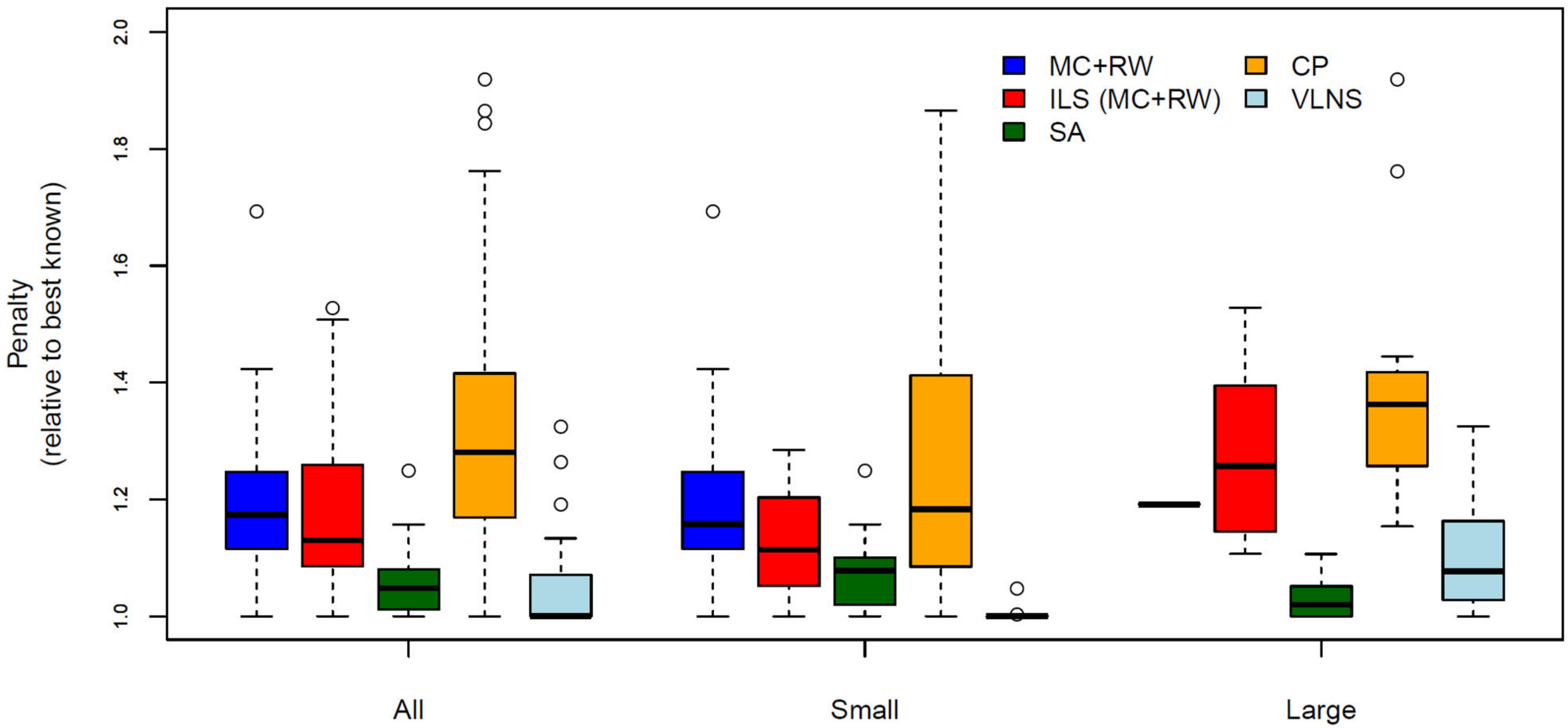
Large neighborhood search



Test Laboratory Scheduling: LNS

Repeatedly generate and solve simplified CP instances:

- Only k projects can be scheduled, the rest of the schedule is fixed
- Initially, $k = 1$, increases when stuck
- Tabu list
- Some scheduling-only steps, with fixed grouping

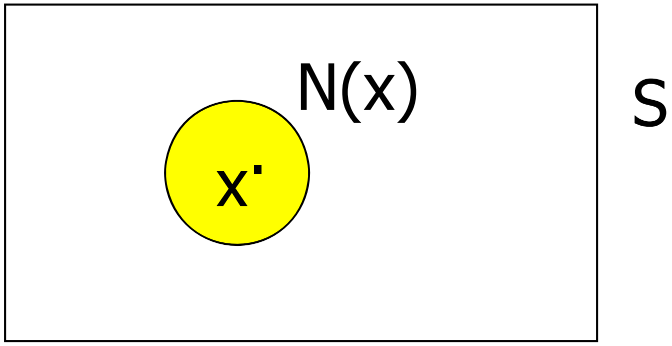


Penalty
(relative to best known)
2.0
1.8
1.6
1.4
1.2
1.0
MC+RW
ILS (MC+RW)
SA
CP
VLNS
All
Small
Large

Metaheuristics

Local Search Techniques

- Based on the neighbourhood of the current solution



- The solution is changed iteratively using neighbourhood relations (moves)

- Acceptable or optimal solutions are often reached

Local Search Techniques

1. Construct the initial solution s
2. Generate neighbourhood N(s) of solution s
3. Select from the neighbourhood the descendant of the current solution
4. Go to step 2

Advanced metaheuristic techniques

- Simulated Annealing
- Tabu Search
- Iterated Local Search
- Min-Conflicts
- …

Metaheuristics include a mechanism to escape local optima

Neighborhoods: Rotating Workforce Scheduing

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
A		D	D	A	D	D	
B		D	A	N	D	N	D
C	D		N	D	D	A	N
D	N				A		A
E	D	N	D			A	D
F	N	A	A	D	A		
G	D	D	A	A	N	N	
H				A	A	D	N
I	A	N					A
J	A	N	N	N			
K	N	A	N	D	N		
L	A	A	D	N	N		

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
A		D	D	A	D	D	
B		D	A	N	D	N	D
C	D		N	D	D	A	N
D	N				A		A
E	D	N	D			A	D
F	N	A	A	D	A		
G	D	D	A	A	N	N	
H				A	A	D	N
I	A	N					A
J	A	N	N	N			
K	N	A	N	D	N		
L	A	A	D	N	N		

Neighborhoods: Test Laboratory Scheduling

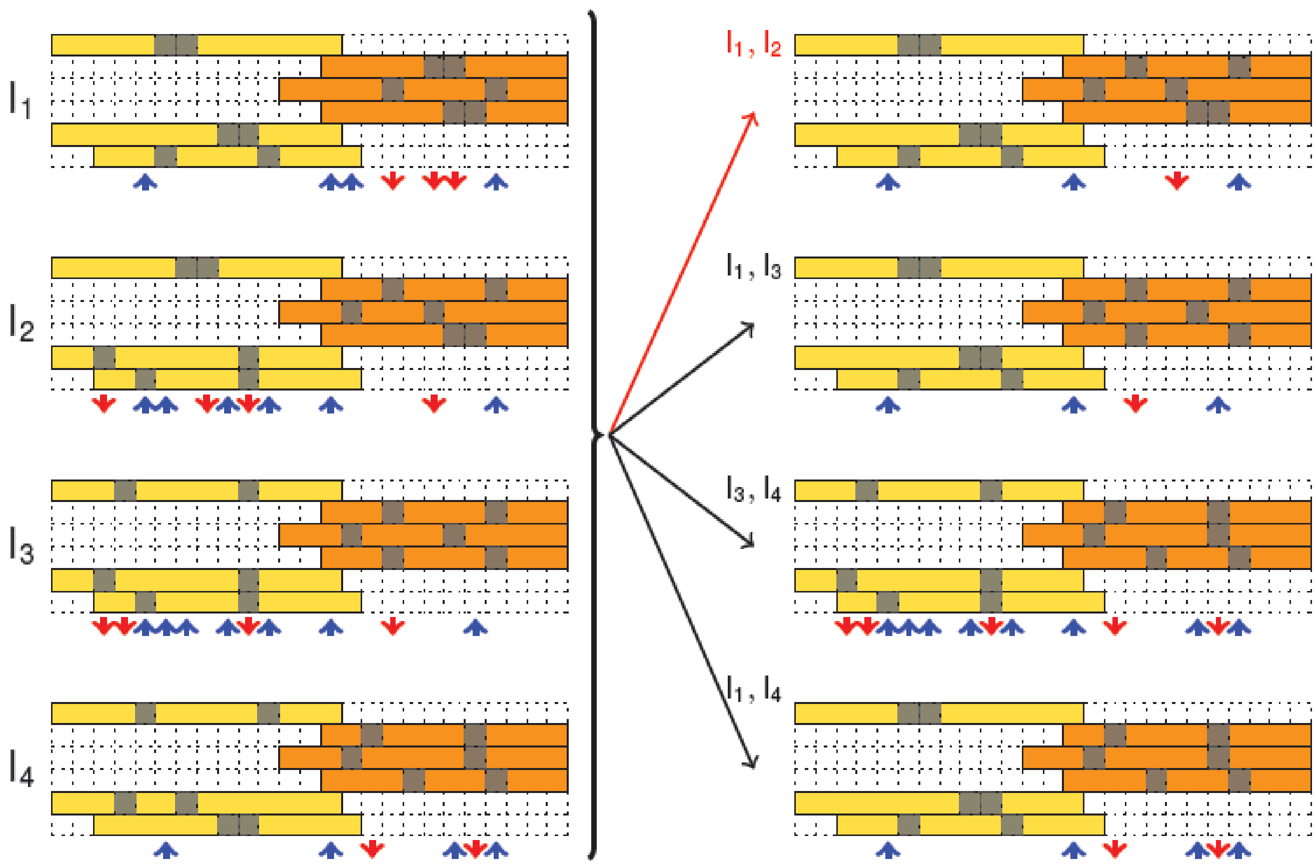
Scheduling neighborhoods

- Timeslot change
- Mode change
- Single resource change
- JobOpt
  - Change all assignments of single job

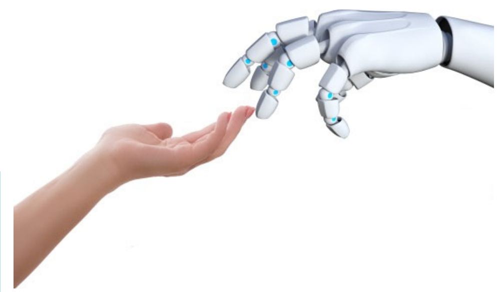
Regrouping neighborhoods

- Transfer task between jobs
- Merge jobs
- Split jobs
  - Move subset of tasks to new job
  - Variant: Linear split

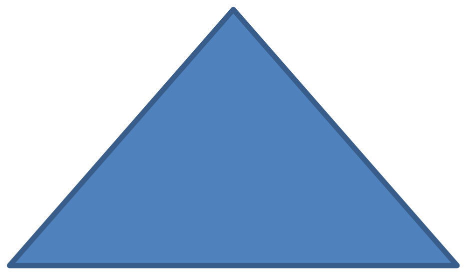
Memetic Algorithms: Crossover



Hybrid techniques



Methods of Artificial Intelligence (Machine Learning, Heuristics…)



Methods of Logic

$$S_{i,d,t} \leftrightarrow \bigwedge_{x=1}^{sl_t} U_{i,d,x} \bigwedge_{y=sl_t}^{sl_{max}} \neg U_{i,d,y}$$

…

Mathematical Optimization

$$\begin{aligned} minimize\ f = \ & 30 * \sum_{\substack{s\in S \\ k\in K \\ d\in\{1\ldots 7\}}} C_{skd}^{S1} \\ & +15 * \sum_{\substack{n\in N \\ s\in S \\ d\in\{1\ldots 7\}}} (C_{nsd}^{S2a} + C_{nsd}^{S2b}) \\ & +30 * \sum_{\substack{n\in N \\ d\in\{1\ldots 7\}}} (C_{nd}^{S2c} + C_{nd}^{S2d}) \end{aligned}$$

Part 1: Conclusions

- Many optimization problems in industry are still solved manually
- AI and optimization offer tremendous potential for further improving solutions in these domains
- Success stories:
  - Test lab scheduling
  - Workforce scheduling
  - Machine scheduling
  - Oven scheduling
  - Educational timetabling, Sport timetabling
  - …
- No free lunch
  - Combination of AI and optimization techniques is crucial

Challenges

- Automated generation of neighborhoods
- Weights for soft constraints
- Explainability
- Automated modeling
- …

Automated algorithm selection

Instance space analysis

Hyper-heuristics

Outlook

Algorithm Selection - Motivation

Often, several search algorithms are available for solving a particular problem

- **No free lunch theorem**
- "...for any algorithm, any elevated performance over one class of problems is offset by performance over another class"
- "...any two algorithms are equivalent when their performance is averaged across all possible problems"

Wolpert and Macready, "No free lunch theorems for optimization", 1997
Wolpert and Macready, "Coevolutionary free lunches", 2005

Algorithm Selection - Motivation

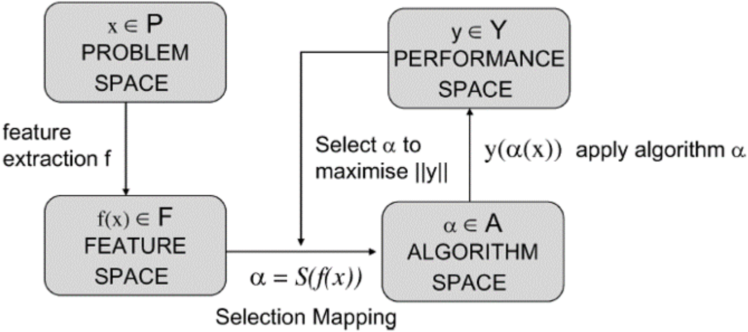
Often, several search algorithms are available for solving a particular problem

- **No free lunch theorem**
- "…for any algorithm, any elevated performance over one class of problems is offset by performance over another class"
- "…any two algorithms are equivalent when their performance is averaged across all possible problems"

⇒ How to select the best algorithm for a specific problem instance?

Wolpert and Macready, "No free lunch theorems for optimization", 1997
Wolpert and Macready, "Coevolutionary free lunches", 2005

Algorithm Selection Problem, Rice (1976)



Rice, "The algorithm selection problem", 1976

Smith-Miles, "Cross-disciplinary perspectives on meta-learning for algorithm selection", 2009

Algorithm Selection Problem, Rice (1976)

Input:

- **Problem space** P that represents the set of instances of a problem class
- **Feature space** F that contains measurable characteristics of the instances generated by a computational feature extraction process applied to P
- Set of considered **algorithms** A for tackling the problem
- **Performance space** Y maps application of an algorithm on an instance to a set of performance metrics

Algorithm Selection Problem: For a given problem instance $x \in P$, with features $f(x) \in F$, find the selection mapping $S(f(x))$ into the algorithm space, such that the selected algorithm $\alpha \in A$ maximizes the performance mapping $y(\alpha(x)) \in Y$.

Back to the Example: Rotating Workforce Scheduling

- Varying demand for different shifts

Shift	Mon	Tue	Wed	Thu	Fri	Sat	Sun
D	1	1	1	1	1	1	1
A	1	1	1	1	1	1	0
N	1	1	1	1	1	1	1

- 4 employees, cyclic schedule
- Regulations constraining shift assignments
- 5-7 days on work, 2-4 days off
- D: 2-5 days, A: 2-4 days, N: 2-3 days
- No D after A or N, no A after N

Back to the Example: Rotating Workforce Scheduling

Problem space P:

- 20 initial real-life instances
- 2000 generated instances

Kletzander et al., "Exact methods for extended rotating workforce scheduling problems", 2019

Musliu, "Heuristic methods for automatic rotating workforce scheduling", 2006

Back to the Example: Rotating Workforce Scheduling

Problem space P:

- 20 initial real-life instances
- 2000 generated instances

Algorithm space A:

- Constraint programming model:
  - MiniZinc modelling language
  - Lazy clause generation solver Chuffed
- Metaheuristic combining methods from:
  - Min-conflict heuristics
  - Tabu search
  - Random walk

Kletzander et al., "Exact methods for extended rotating workforce scheduling problems", 2019

Musliu, "Heuristic methods for automatic rotating workforce scheduling", 2006

Back to the Example: Rotating Workforce Scheduling

Performance space Y:

- Satisfaction problem
- Measure runtime to feasible solution (timeout 1000 seconds)

Back to the Example: Rotating Workforce Scheduling

Performance space Y:

- Satisfaction problem
- Measure runtime to feasible solution (timeout 1000 seconds)

Feature space F: How to get features from instance data?

- n employees
- Length of schedule w
- Set of work shifts $\mathbf{A}$ + day off O, $\mathbf{A}^+ = \mathbf{A} \cup \{O\}$
- Temporal requirement matrix R
- Min and max work block length ℓ_w and u_w
- Min and max block lengths for shifts and days off ℓ_s and u_s ($s \in \mathbf{A}^+$)
- Set of forbidden sequences $\mathbf{F}$

Direct Instance Features

Take instance data to directly use as features:

- Number of employees n
- Number of shifts m
- Minimum and maximum length of work blocks ℓ_w and u_w as well as blocks off shift ℓ_O and u_O.
- Minimum, maximum and average for each of the sets $\{\ell_s \mid s \in \mathbf{A}\}$ and $\{u_s \mid s \in \mathbf{A}\}$.
- Number of forbidden sequences f.

Advanced Instance Features

Compute features from relations, matrices, graphs, ...

- *workFraction*: Percentage of all days spent working
- *shiftFraction*: Distribution of requirements between shifts
- *blockTightness*: $blockTightness = up - low$
- *avgBlockLength*: Lower and upper bound for the average block length
- *shiftBlockTightness*: Freedom in choosing block lengths for individual shift types
- *shiftDayFactor*: Regularity of shifts throughout the week
- *dayFraction*: Workload in relation to the number of employees for individual days
- *dailyChange*: Change in workload between consecutive days

Model Features

Run fast algorithm initializations, heuristics, ...

- MiniZinc to FlatZinc conversion statistics
  - Number of boolean and interger variables
  - Number of boolean and integer constraints
- Initialization in Chuffed:
  - Number of variables, propagators, SAT variables
  - Number of binary, ternary, and long clauses
  - Average length of long clauses

Algorithm Selection

Use any supervised machine learning approach of your choice:

- Bayesian Networks
- Decision Trees
- k-Nearest Neighbor
- Random Forests
- Multilayer Perceptrons
- Support Vector Machines
- Deep Neural Networks

Algorithm Selection and Analysis for RWS

- Method: Random Forests
- Chuffed vs. metaheuristic: accuracy 80%
- Predict timeout: accuracy 93%
- Feasible vs. infeasible: accuracy 98%
- Regression on magnitude of runtime: correlation 0.7 to 0.8

Learning within Algorithms

In this tutorial section: Decision between different algorithms

Other option: Selection / learning within algorithms

- Later in this tutorial: Learning to select algorithm components (hyper-heuristics)
- Example for tree search: Variable / value selection

Learning without Features

Finding adequate features is one of the main challenges in algorithm selection

$\Rightarrow$ What about algorithm selection without features?

- Recent research direction
- Directly use instance data as time series for Recurrent Neural Network (RNN)
- Application to online 1D bin packing

Alissa, Sim, and Hart, "Automated algorithm selection: from feature-based to feature-free approaches", 2023

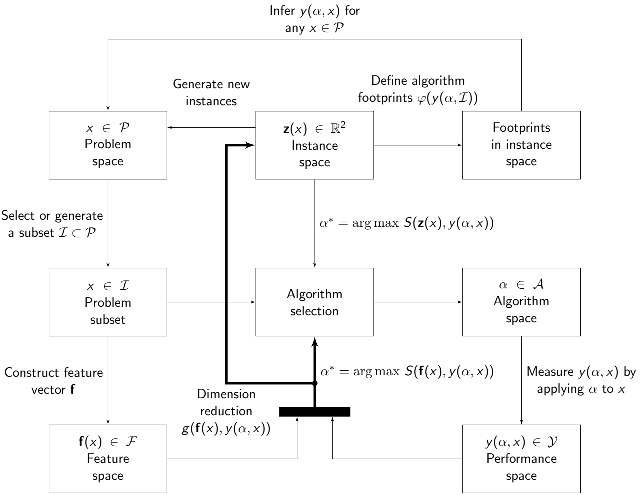
Instance Space Analysis - Motivation

How do we analyze which method works well on which instances?
How do we evaluate a new method for our problem?

- Use benchmark instances
- Better in the average?
- Better in certain cases?
- Do the benchmark instances cover all interesting areas?

$\Rightarrow$ How to check instances and features to make sure that we can properly identify strengths and weaknesses of different algorithms?

Extending Rice's Framework, Smith-Miles et. al. (2014)



Smith-Miles et al., "Towards objective measures of algorithm performance across instance space", 2014

Extending Rice's Framework, Smith-Miles et. al. (2014)

Extensions to Rice's framework:

- Separation of **Problem space** P and available **sub-space of instances** I
- **2-dimensional instance space** for visualization of instance and features distributions
- Selection mapping can either be computed from the feature space or from the instance space
- Performance can be visualized in the instance space and inferred for unseen instances

Instance Space Analysis

Goals:

- Visualize distribution and diversity of instances
- Assess adequacy of features
- Identify regions of strength **footprints** and weaknesses
- Infer where additional instances might be needed

Smith-Miles and Muñoz, "Instance Space Analysis for Algorithm Testing: Methodology and Software Tools", 2023

Instance Space Analysis

Goals:

- Visualize distribution and diversity of instances
- Assess adequacy of features
- Identify regions of strength **footprints** and weaknesses
- Infer where additional instances might be needed

Software Tool: MATILDA

`https://matilda.unimelb.edu.au/matilda/`

`https://github.com/andremun/InstanceSpace`

Smith-Miles and Muñoz, "Instance Space Analysis for Algorithm Testing: Methodology and Software Tools", 2023

Back to the Example: Rotating Workforce Scheduling

Sub-space of instances I:

- 20 initial real-life instances
- 2000 generated instances

Kletzander et al., "Exact methods for extended rotating workforce scheduling problems", 2019

Musliu, "Heuristic methods for automatic rotating workforce scheduling", 2006

Back to the Example: Rotating Workforce Scheduling

Sub-space of instances I:

- 20 initial real-life instances
- 2000 generated instances

Algorithm space A:

- **2 constraint programming models**:
  - Model 2 extends model 1 by additional constraint to check sequences at the start of each block
- Metaheuristic

Same performance space Y (runtime) and feature space F

Kletzander et al., "Exact methods for extended rotating workforce scheduling problems", 2019

Musliu, "Heuristic methods for automatic rotating workforce scheduling", 2006

Original Projection

- Bound extreme outliers
- Normalization using Box-Cox and Z transformation
- Remove low diversity features
- Retain features with high correlation to performance
- Clustering

Kletzander, Musliu, and Smith-Miles, "Instance space analysis for a personnel scheduling problem", 2021

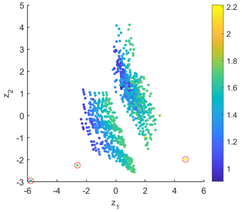
Original Projection

- Bound extreme outliers
- Normalization using Box-Cox and Z transformation
- Remove low diversity features
- Retain features with high correlation to performance
- Clustering

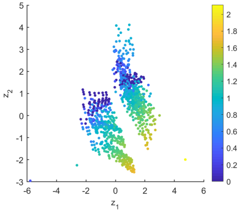
$$\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} -0.45 & -0.39 \\ 0.45 & 0.40 \\ 0.50 & 0.08 \\ -0.32 & 0.37 \\ 0.23 & -0.63 \end{pmatrix}^{\intercal} \cdot \begin{pmatrix} maxShiftDayFactor' \\ maxDayFraction' \\ employees' \\ minAvgBlockLength' \\ blockTightness' \end{pmatrix}$$

Kletzander, Musliu, and Smith-Miles, "Instance space analysis for a personnel scheduling problem", 2021

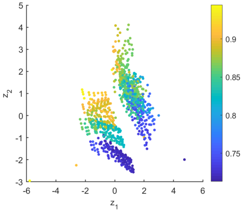
Original Feature Distribution



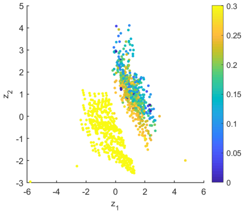
employees



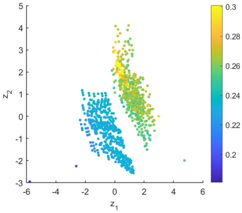
blockTightness



minAvgBlockLength



maxShiftDayFactor



maxDayFraction

Original Feature Distribution

- Good visualization of feature distribution
- Most influential features:
  - Possible block length distributions (*blockTightness*, *minAvgBlockLength*)
  - Instance size (*employees*)
  - Distribution throughout the week (*maxShiftDayFactor*)
  - Daily workload (*maxDayFraction*)
- 2 separated visible clusters
- Several real-life instances are outliers

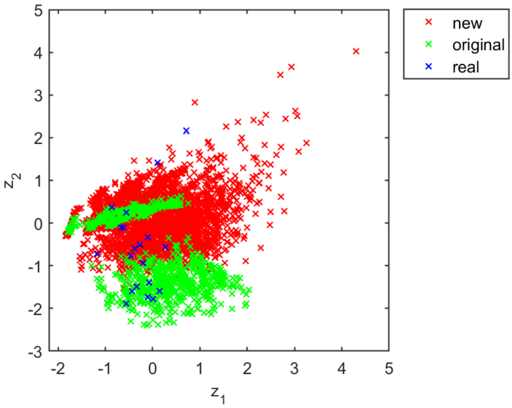
Original Feature Distribution

- Good visualization of feature distribution
- Most influential features:
  - Possible block length distributions (*blockTightness*, *minAvgBlockLength*)
  - Instance size (*employees*)
  - Distribution throughout the week (*maxShiftDayFactor*)
  - Daily workload (*maxDayFraction*)
- 2 separated visible clusters
- Several real-life instances are outliers

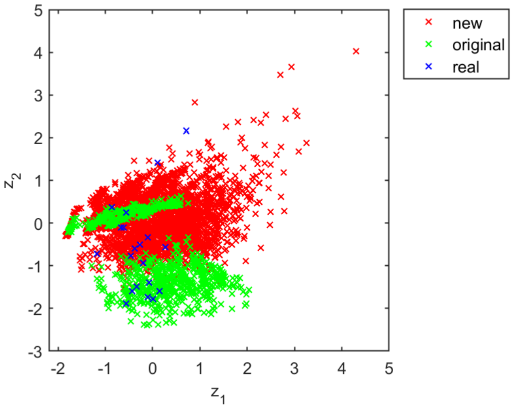
Analysis indicates more instances would be beneficial

- Adapt instance generator
  - Cover gap
  - Include real-life instances
  - Increase number of employees
- Added 3480 new instances

Extended Instances

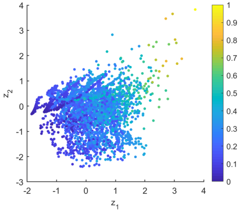


Extended Instances

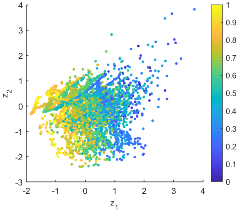


$$\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} -0.31 & 0.31 \\ 0.02 & -0.57 \\ -0.47 & -0.08 \\ 0.44 & 0.15 \end{pmatrix}^{\mathsf{T}} \cdot \begin{pmatrix} minDayFraction' \\ maxDayFraction' \\ maxAvgBlockLength' \\ minAvgBlockLength' \end{pmatrix}$$

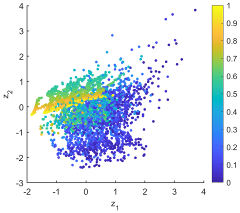
Extended Instance Set - Feature Distribution



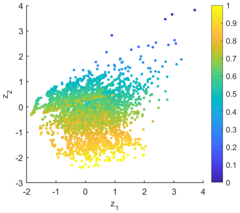
minAvgBlockLength



maxAvgBlockLength



minDayFraction

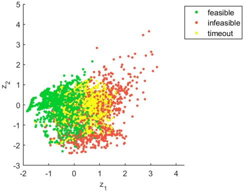


maxDayFraction

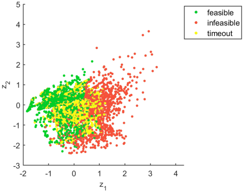
Extended Instance Set - Feature Distribution

- z_1: Axis for *avgBlockLength*
  - Low minimum and high maximum on the left
  - High minimum and low maximum on the right
- z_2: Axis for *dayFraction*
  - Low minimum and high maximum on the bottom
  - High minimum and low maximum on the top
- Gap is closed and real-life instances are well covered

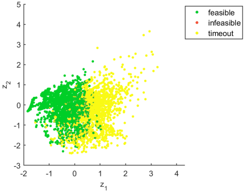
Algorithm Results - Feasibility



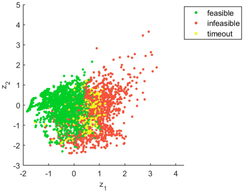
Chuffed model 1



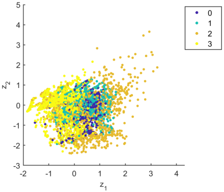
Chuffed model 2



Metaheuristic

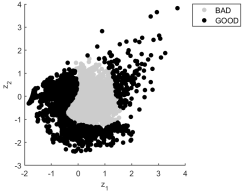


All methods combined

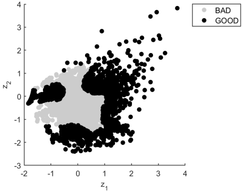


Number of results

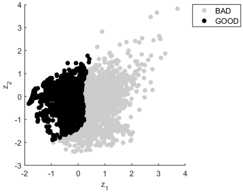
Algorithm Results - Footprints



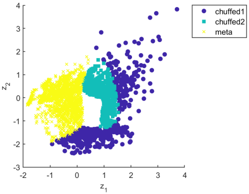
Chuffed model 1



Chuffed model 2



Metaheuristic



SVM portfolio

Algorithm Results

- Clearly visible boundaries between feasibility and infeasibility
  - Due to bounds for number of blocks on z_1-axis
  - Due to high demand fluctuations on z_2-axis
- Instances along this boundary are most difficult

- Strong and weak areas can be generalized to footprints
- Algorithm portfolio can be calculated from instance space
  - Recommended algorithm for each instance
  - Generalization to further areas can be attempted
  - Some areas might not have any well-performing algorithms $\rightarrow$ can be reported as hard to solve

$\Rightarrow$ Instance Space Analysis allows deep insights in algorithm behaviour and instance distribution

Automated algorithm selection

Instance space analysis

Hyper-heuristics

- CHeSC
- Reinforcement learning
- Real-world problem domains
- Example: Online Bin Packing

Outlook

Example: CP

- Modern CP solvers internally employ heuristics
- Large Neighborhood Search (LNS):
  Repeatedly apply partial relaxation, then reconstruct

Thomas and Schaus, "Revisiting the Self-adaptive Large Neighborhood Search", 2018

Example: CP

- Modern CP solvers internally employ heuristics
- Large Neighborhood Search (LNS):
  Repeatedly apply partial relaxation, then reconstruct

Relaxation

Random $x\%$ of variables are relaxed

Propagation Guided Fix groups of dependent variables

Value Guided Relax variables with same value

Precedency based Assume values are start times, build partial random order

...

Thomas and Schaus, "Revisiting the Self-adaptive Large Neighborhood Search", 2018

Example: CP

- Modern CP solvers internally employ heuristics
- Large Neighborhood Search (LNS):
  Repeatedly apply partial relaxation, then reconstruct

Relaxation

Random x% of variables are relaxed

Propagation Guided Fix groups of dependent variables

Value Guided Relax variables with same value

Precedency based Assume values are start times, build partial random order

...

Reconstruction

Limited backtracking search

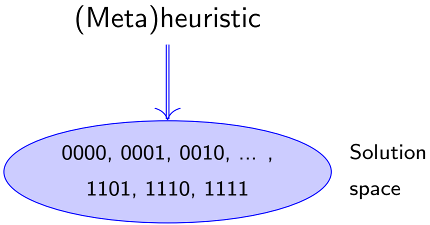
Variable selection:
First Fail, Most Recent Conflict, Weighted Degree

Value selection:
Min/max domain, random, value sticking,...

Thomas and Schaus, "Revisiting the Self-adaptive Large Neighborhood Search", 2018

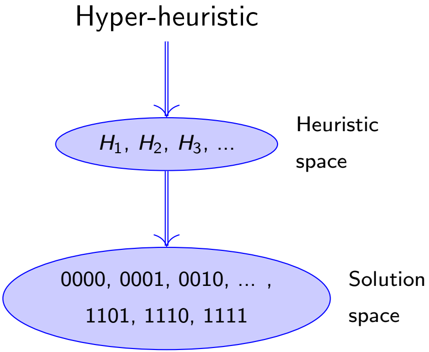
(Meta)heuristic approach

- Operates on set of (possible) solutions
- Implementation defines sample order

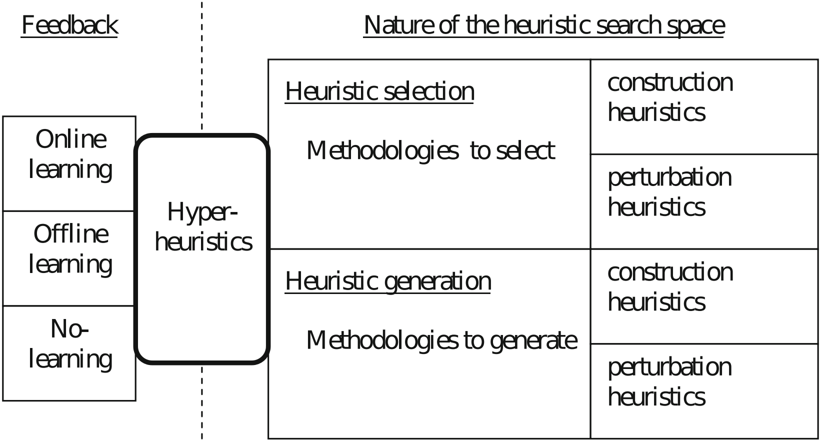


Hyper-heuristic approach

- Operates on set of (low-level) heuristics
  - Complete algorithms
  - Algorithmic components
- Indirectly explore solution space via low-level heuristics

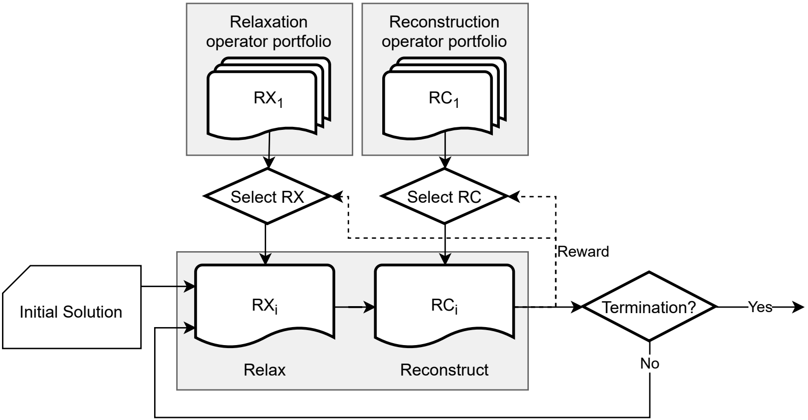


Classification



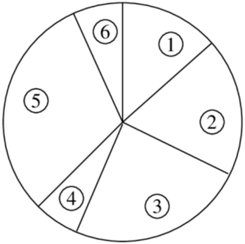
Source: Burke et al., "A Classification of Hyper-Heuristic Approaches: Revisited", 2019

Example: CP - Adaptive Large Neighborhood Search



Example: CP - Operator selection

- Assign weight to each operator
- Select relaxation and reconstruction operator based on current weight (Roulette Wheel Selection)

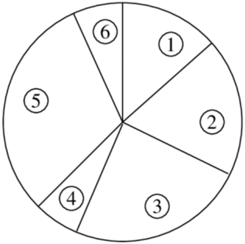


Laborie and Godard, "Self-adapting large neighborhood search: Application to single-mode scheduling problems", 2007

Thomas and Schaus, "Revisiting the Self-adaptive Large Neighborhood Search", 2018

Example: CP - Operator selection

- Assign weight to each operator
- Select relaxation and reconstruction operator based on current weight (Roulette Wheel Selection)



- Update weights according to result:

$$weight_{t+1}(o) = (1 - \alpha) * weight_t(o) + \alpha * \frac{\Delta c}{\Delta t}$$

Laborie and Godard, "Self-adapting large neighborhood search: Application to single-mode scheduling problems", 2007

Thomas and Schaus, "Revisiting the Self-adaptive Large Neighborhood Search", 2018

Example: CP - Results

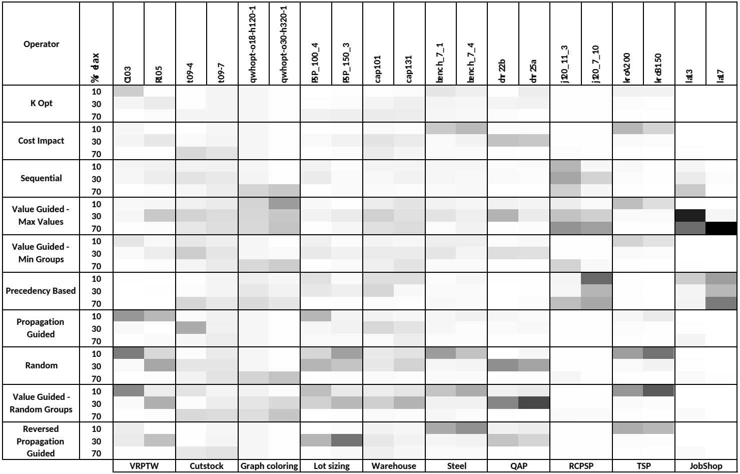


Fig. 1. Heat map of the relaxation operators selection for the Eval window approach

Thomas and Schaus, "Revisiting the Self-adaptive Large Neighborhood Search", 2018

Cross-Domain Heuristic Search Challenge

- Proposed in 2011[1]
- 6 problem domains:
  - Max-SAT, Bin Packing, Personnel Scheduling, Flow Shop, TSP, VRP

[1] Ochoa et al., "HyFlex: A Benchmark Framework for Cross-Domain Heuristic Search", 2012

Cross-Domain Heuristic Search Challenge

- Proposed in 2011[1]
- 6 problem domains:
  - Max-SAT, Bin Packing, Personnel Scheduling, Flow Shop, TSP, VRP
- Domain implementations and instance data hidden from hyper-heuristics

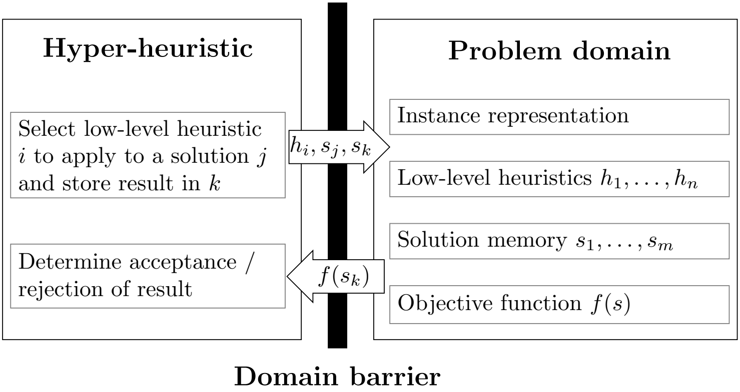
[1] Ochoa et al., "HyFlex: A Benchmark Framework for Cross-Domain Heuristic Search", 2012

Cross-Domain Heuristic Search Challenge

- Proposed in 2011[1]
- 6 problem domains:
  - Max-SAT, Bin Packing, Personnel Scheduling, Flow Shop, TSP, VRP
- Domain implementations and instance data hidden from hyper-heuristics
- Introduced hyper-heuristic framework HyFlex

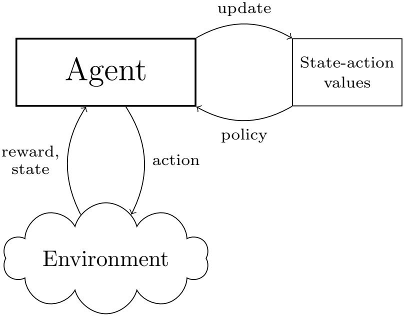
[1] Ochoa et al., "HyFlex: A Benchmark Framework for Cross-Domain Heuristic Search", 2012

HyFlex



Hyper-heuristic
Select low-level heuristic i to apply to a solution j and store result in k
Determine acceptance / rejection of result
h_i, s_j, s_k
$f(s_k)$
Problem domain
Instance representation
Low-level heuristics $h_1, \ldots, h_n$
Solution memory $s_1, \ldots, s_m$
Objective function $f(s)$
Domain barrier

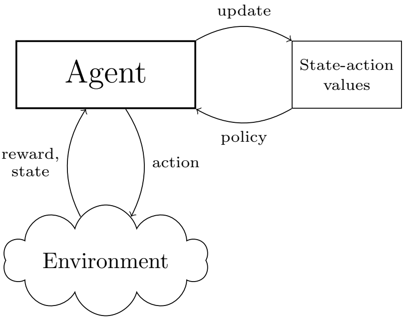
Reinforcement learning



Mischek and Musliu, "Reinforcement Learning for Cross-Domain Hyper-Heuristics", 2022

Kletzander and Musliu, "Large-State Reinforcement Learning for Hyper-Heuristics", 2023

Reinforcement learning



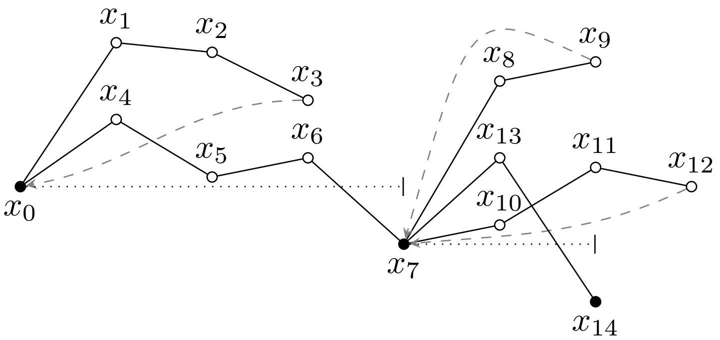
- Natural fit
  - Actions: low-level heuristics
  - Reward: Function of objective value
- Different options for remaining components:
  - State representation
  - Decision policy
  - Update rule

Mischek and Musliu, "Reinforcement Learning for Cross-Domain Hyper-Heuristics", 2022

Kletzander and Musliu, "Large-State Reinforcement Learning for Hyper-Heuristics", 2023

RL - Solution chains

- Periodically reset solution, if no improvement found
- Balance long, expensive chains with short chains of limited reach
  - Best results following Luby's sequence



Chuang and Smith, "A study of agnostic hyper-heuristics based on sampling solution chains", 2017

RL - State representation

- Issue: Most interesting information is hidden
- Intuition: Extract information from search history and trajectory of objective value

RL - State representation

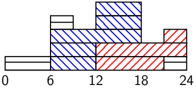
- Issue: Most interesting information is hidden
- Intuition: Extract information from search history and trajectory of objective value

- Last heuristic
- Last heuristic type
- Last change sign
- Last change magnitude
- Chain progress
- Steps since last improvement magnitude
- Steps magnitude and time
- Objective relative to initial or best
- Relative number of improving / 0-cost heuristics
- Measures of recent heuristics
- ...

Problem-independent hyper-heuristics on new domains

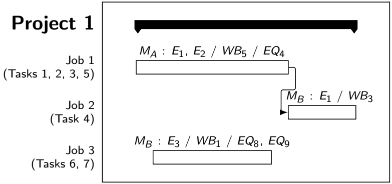
Empl.	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	D	D	D	D	N	N	-
2	-	-	A	A	A	A	N
3	N	N	-	-	D	D	D
4	A	A	N	N	-	-	-

Rotating Workforce Schedule

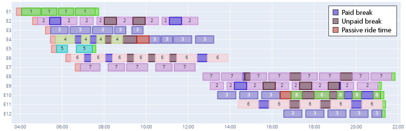


Start	End	E.
6:00	18:00	3
12:00	24:00	2
21:00	9:00	1

Minimum Shift Design



Test Laboratory Scheduling



Bus Driver Scheduling

TLSP: Low-level-heuristic portfolio

Mutation

- Random move: Mode, time, resources, grouping
- Randomize jobs
- Random walk

TLSP: Low-level-heuristic portfolio

Mutation

- Random move: Mode, time, resources, grouping
- Randomize jobs
- Random walk

Ruin and recreate

- Delete and reschedule
- Delete and regroup

TLSP: Low-level-heuristic portfolio

Mutation

- Random move: Mode, time, resources, grouping
- Randomize jobs
- Random walk

Ruin and recreate

- Delete and reschedule
- Delete and regroup

Crossover

- Random projects
- Single point XO
- Two point XO

TLSP: Low-level-heuristic portfolio

Mutation

- Random move: Mode, time, resources, grouping
- Randomize jobs
- Random walk

Ruin and recreate

- Delete and reschedule
- Delete and regroup

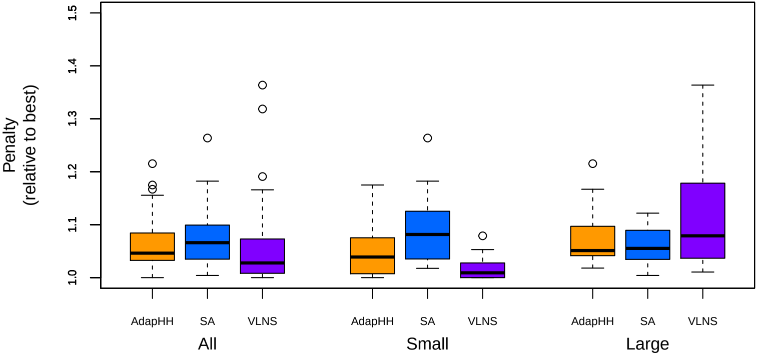
Crossover

- Random projects
- Single point XO
- Two point XO

Local search

- HillClimbing
  - mode & time, resources, JobOpt, grouping
- MinConflict
  - mode & time, resources, JobOpt, grouping
- Stochastic hill climbing
  - all neighborhoods
  - high, medium, low T
- Single project CP
- Job-wise greedy

Experimental results: TLSP



Mischek and Musliu, "Leveraging problem-independent hyper-heuristics for real-world test laboratory scheduling", 2023

Experimental results: Bus Driver Scheduling

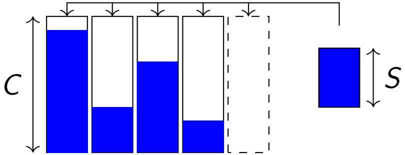
Instance	SA	CH-FR	CH-PR	GIHH	L-GIHH	LAST-RL
10	**14717.4**	14838.8	14805.6	14787.0	14773.6	14779.8
20	30860.6	30706.6	30671.2	30731.6	30694.0	**30669.4**
30	50947.4	50946.6	50903.6	**50765.8**	50854.2	50890.0
40	69119.8	68583.4	68847.6	68639.6	68645.4	**68478.2**
50	87013.2	87091.2	87034.0	86762.0	86729.8	**86681.8**
60	103967.6	103521.8	103464.8	103138.8	103149.8	**102935.8**
70	122753.6	122247.2	122025.6	121671.8	**121660.6**	121916.2
80	140482.4	139382.4	139209.2	139123.0	**139041.6**	139250.2
90	156385.0	154938.0	154972.4	155093.8	155113.2	**154915.0**
100	173524.0	171718.6	**171182.4**	171278.2	171325.4	171589.4

Kletzander and Musliu, "Hyper-Heuristics for Personnel Scheduling Domains", 2022

Online Bin Packing

Goal: pack sequence of items in as few bins as possible

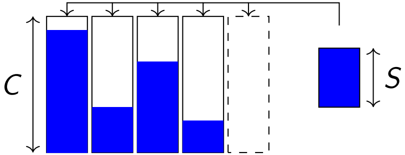
- Fixed capacity C for bins
- Items packed one-by-one
- Size of future items unknown



Online Bin Packing

Goal: pack sequence of items in as few bins as possible

- Fixed capacity C for bins
- Items packed one-by-one
- Size of future items unknown



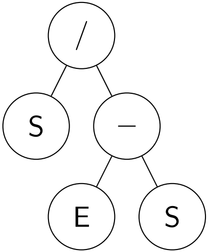
Popular heuristic: *Best Fit* - Choose (feasible) bin with smallest capacity

Online Bin Packing - Genetic Programming

- Compute score for each bin per item
- Assign to bin with highest score

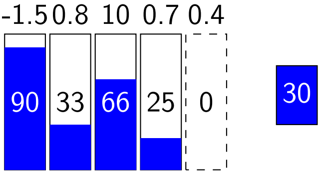
Burke et al., "The scalability of evolved on line bin packing heuristics", 2007

Online Bin Packing - Genetic Programming



- Compute score for each bin per item
- Assign to bin with highest score

- Evaluation tree
- Functions: +, -, *, /
- Terminals: S, E (*emptiness*, remaining capacity)



Burke et al., "The scalability of evolved on line bin packing heuristics", 2007

Online Bin Packing - Genetic Programming

- Heuristics evolved on sequences of 100 - 500 items
- Evaluated on much longer sequences (up to 100000)
- Best Fit better up to half size of training sequences, then evolved heuristics take the lead

Burke et al., "The scalability of evolved on line bin packing heuristics", 2007

Tauritz and Woodward, "Generative Hyper-Heuristics", 2022

Online Bin Packing - Genetic Programming

- Heuristics evolved on sequences of 100 - 500 items
- Evaluated on much longer sequences (up to 100000)
- Best Fit better up to half size of training sequences, then evolved heuristics take the lead
- Intuition: Item of size 20 may fit in gap of size 30, but better item (size 25-30) is likely to come along eventually.

Burke et al., "The scalability of evolved on line bin packing heuristics", 2007

Tauritz and Woodward, "Generative Hyper-Heuristics", 2022

Online Bin Packing - Genetic Programming

- Heuristics evolved on sequences of 100 - 500 items
- Evaluated on much longer sequences (up to 100000)
- Best Fit better up to half size of training sequences, then evolved heuristics take the lead
- Intuition: Item of size 20 may fit in gap of size 30, but better item (size 25-30) is likely to come along eventually.

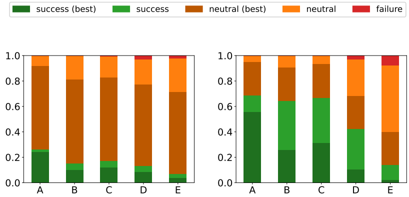
Other applications: Black-box search operators, graph partitioning, graph generation, ...

Burke et al., "The scalability of evolved on line bin packing heuristics", 2007

Tauritz and Woodward, "Generative Hyper-Heuristics", 2022

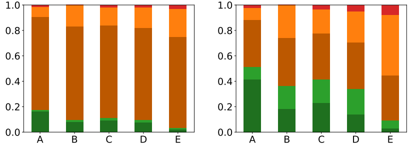
Preference Explanation and Decision Support for Multi-Objective Real-World Test Laboratory Scheduling

- Preference weights for multi-objective problems can be challenging to determine
- Shapley values can be used to capture relationships between objectives and provide useful suggestions for weight updates
- Case study: Decision support system for multi-objective TLSP



(a) *uniform*, $\gamma = 1.5$

(b) *uniform*, $\gamma = 10$

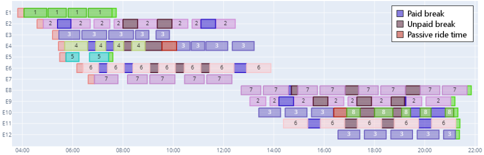


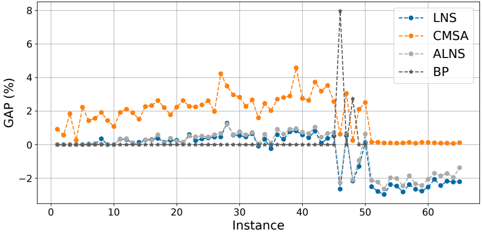
(c) *exponential*, $\gamma = 1.5$

(d) *exponential*, $\gamma = 10$

Investigating Large Neighbourhood Search for Bus Driver Scheduling

- Hybrid solution method for complex real-life scheduling problem
- Select meaningful subproblem based on problem structure
- Solve subproblem (almost) exactly using Column Generation





Co-Authors/Selected References

1) Philipp Danzinger, Tobias Geibinger, David Janneau, Florian Mischek, Nysret Musliu, Christian Poschalko: A System for Automated Industrial Test Laboratory Scheduling. ACM Trans. Intell. Syst. Technol. 14(1): 3:1-3:27 (2023)
2) Lucas Kletzander, Nysret Musliu: Solving the general employee scheduling problem. Comput. Oper. Res. 113 (2020)
3) Nysret Musliu, Andrea Schaerf, Wolfgang Slany: Local search for shift design. Eur. J. Oper. Res. 153(1): 51-64 (2004)
4) Andreas Beer, Johannes Gärtner, Nysret Musliu, Werner Schafhauser, Wolfgang Slany: An AI-Based Break-Scheduling System for Supervisory Personnel. IEEE Intell. Syst. 25(2): 60-73 (2010)
5) Florian Mischek, Nysret Musliu: A local search framework for industrial test laboratory scheduling. Ann. Oper. Res. 302(2): 533-562 (2021)
6) Felix Winter, Nysret Musliu: Constraint-based Scheduling for Paint Shops in the Automotive Supply Industry. ACM Trans. Intell. Syst. Technol. 12(2): 17:1-17:25 (2021)
7) Felix Winter, Nysret Musliu, Emir Demirovic, Christoph Mrkvicka: Solution Approaches for an Automotive Paint Shop Scheduling Problem. ICAPS 2019: 573-581
8) Marie-Louise Lackner, Christoph Mrkvicka, Nysret Musliu, Daniel Walkiewicz, Felix Winter: Minimizing Cumulative Batch Processing Time for an Industrial Oven Scheduling Problem. CP 2021: 37:1-37:18 and Constraint Journal (2023)
9) Martin Josef Geiger, Lucas Kletzander, Nysret Musliu: Solving the Torpedo Scheduling Problem. *Journal of Artificial Intelligence Research. Vol 66: 1-32, 2019*
10) Maximilian Moser, Nysret Musliu, Andrea Schaerf, Felix Winter: Exact and metaheuristic approaches for unrelated parallel machine scheduling. J. Sched. 25(5): 507-534 (2022)
11) Nysret Musliu, Andreas Schutt, Peter J. Stuckey: Solver Independent Rotating Workforce Scheduling. CPAIOR 2018: 429-445
12) Nysret Musliu, Johannes Gärtner, Wolfgang Slany:Efficient generation of rotating workforce schedules. Discret. Appl. Math. 118(1-2): 85-98 (2002)
13) Lucas Kletzander, Nysret Musliu, Johannes Gärtner, Thomas Krennwallner, Werner Schafhauser: Exact Methods for Extended Rotating Workforce Scheduling Problems. ICAPS 2019: 519-527

Co-Authors/Selected References

14) Nysret Musliu: Combination of Local Search Strategies for Rotating Workforce Scheduling Problem. IJCAI 2005: 1529-1530
15) Michael Abseher, Nysret Musliu, Stefan Woltran: Improving the Efficiency of Dynamic Programming on Tree Decompositions via Machine Learning. J. Artif. Intell. Res. 58: 829-858 (2017)
16) Magdalena Widl, Nysret Musliu:The break scheduling problem: complexity results and practical algorithms. Memetic Comput. 6(2): 97-112 (2014)
17) Simon Strassl, Nysret Musliu: Instance space analysis and algorithm selection for the job shop scheduling problem. Comput. Oper. Res. 141: 105661 (2022)
18) Arnaud De Coster, Nysret Musliu, Andrea Schaerf, Johannes Schoisswohl, Kate Smith-Miles: Algorithm selection and instance space analysis for curriculum-based course timetabling. J. Sched. 25(1): 35-58 (2022)
19) Lucas Kletzander, Nysret Musliu, Kate Smith-Miles: Instance space analysis for a personnel scheduling problem. Ann. Math. Artif. Intell. 89(7): 617-637 (2021)
20) Lucas Kletzander, Nysret Musliu: Hyper-Heuristics for Personnel Scheduling Domains. ICAPS 2022: 462-470
21) Florian Mischek, Nysret Musliu: Reinforcement Learning for Cross-Domain Hyper-Heuristics. IJCAI 2022: 4793-4799
22) Lucas Kletzander and Nysret Musliu. Large-state reinforcement learning for hyper-heuristics. Proceedings of the 37th AAAI Conference on Artificial Intelligence, 2023.
23) Michael Abseher, Nysret Musliu, Stefan Woltran:Improving the Efficiency of Dynamic Programming on Tree Decompositions via Machine Learning. J. Artif. Intell. Res. 58: 829-858 (2017)
24) Emir Demirovic, Nysret Musliu: MaxSAT-based large neighborhood search for high school timetabling. Comput. Oper. Res. 78: 172-180 (2017)
25) Nysret Musliu, Felix Winter: A Hybrid Approach for the Sudoku Problem: Using Constraint Programming in Iterated Local Search. IEEE Intell. Syst. 32(2): 52-62 (2017)
26) Markus Triska, Nysret Musliu: An effective greedy heuristic for the Social Golfer Problem. Ann. Oper. Res. 194(1): 413-425 (2012)