A Permutation-Based Heuristic for the Blocking Job Shop Problem

Julia Lange[1] Frank Werner[2]

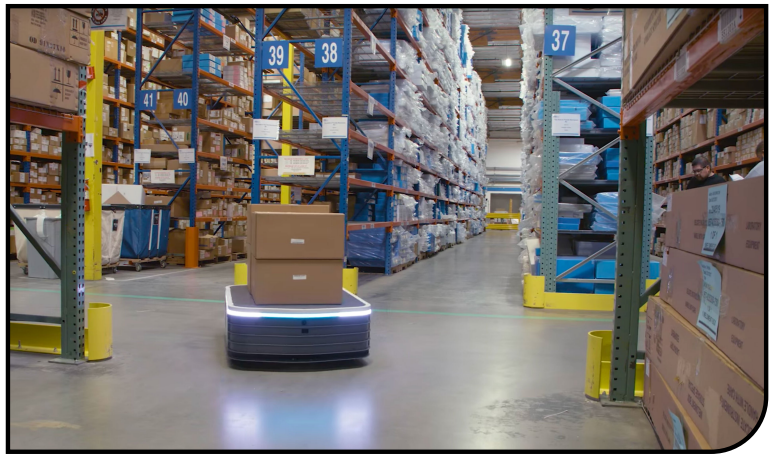
[1] FZI Research Center for Information Technology, Karlsruhe, (lange@fzi.de)

[2] Otto-von-Guericke-University, Magdeburg, (frank.werner@ovgu.de)

IFAC Conference on Manufacturing Modeling, Management, and Control

Berlin, August 28-30, 2019

Scheduling Problems



The problem

$$Jm \mid r_i, block, recrc \mid \sum T_i$$

is defined by

- a set of n **jobs** $J_i \in \mathcal{J}$,
- a set of m **machines** $M_k \in \mathcal{M}$ and
- an individual **technological route** for each job
  $\mathrm{TR}_i : M_{k_1} \rightarrow M_{k_2} \rightarrow \cdots \rightarrow M_{k_{n_i}}$.
- A job consists of an ordered set of **operations** $O_{i,j}$, $j = 1, \ldots, n_i$,
  with **processing times** $p_{i,j}$.
- Recirculation $(recrc)$ is allowed.
- Every job has a given **release time** r_i and a desired **due date** d_i.
- Blocking constraints $(block)$ exist for intermediate processing steps.

The problem

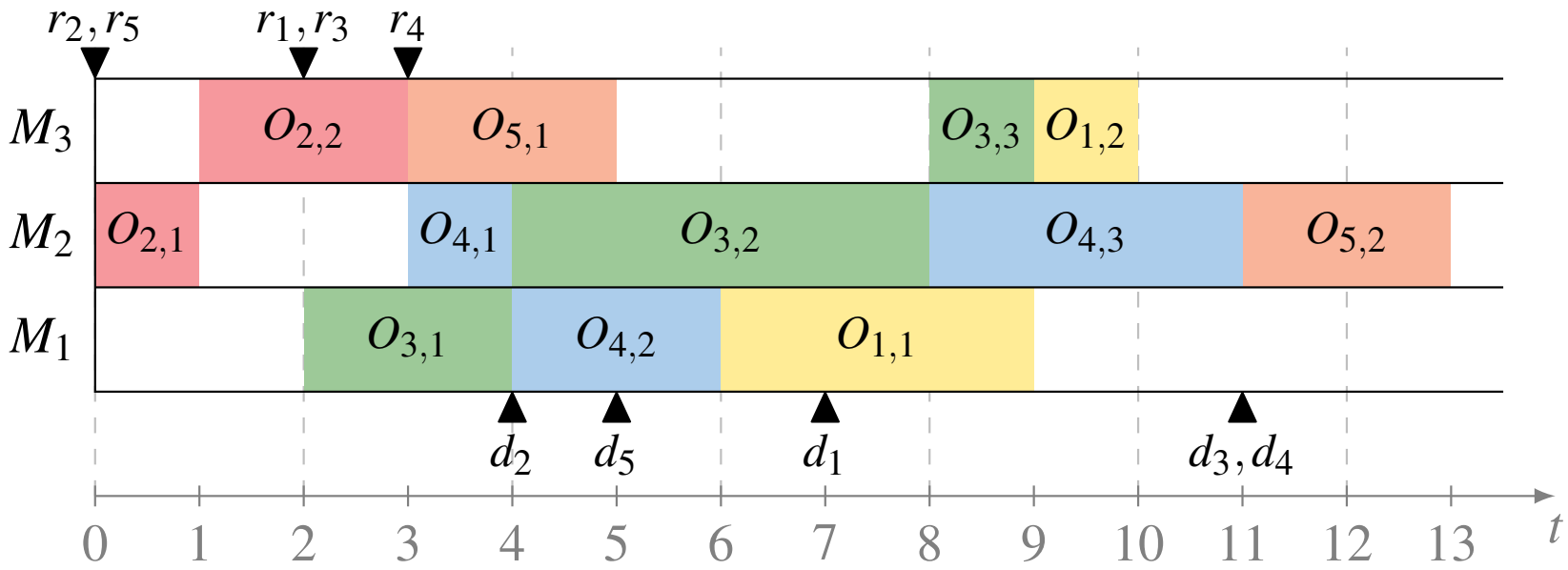
$$Jm \mid r_i, block, recrc \mid \sum T_i$$

is defined by

- a set of n **jobs** $J_i \in \mathcal{J}$,
- a set of m **machines** $M_k \in \mathcal{M}$ and
- an individual **technological route** for each job $\mathrm{TR}_i : M_{k_1} \to M_{k_2} \to \cdots \to M_{k_{n_i}}$.
- A job consists of an ordered set of **operations** $O_{i,j}$, $j = 1, \ldots, n_i$, with **processing times** $p_{i,j}$.
- Recirculation $(recrc)$ is allowed.
- Every job has a given **release time** r_i and a desired **due date** d_i.
- Blocking constraints $(block)$ exist for intermediate processing steps.

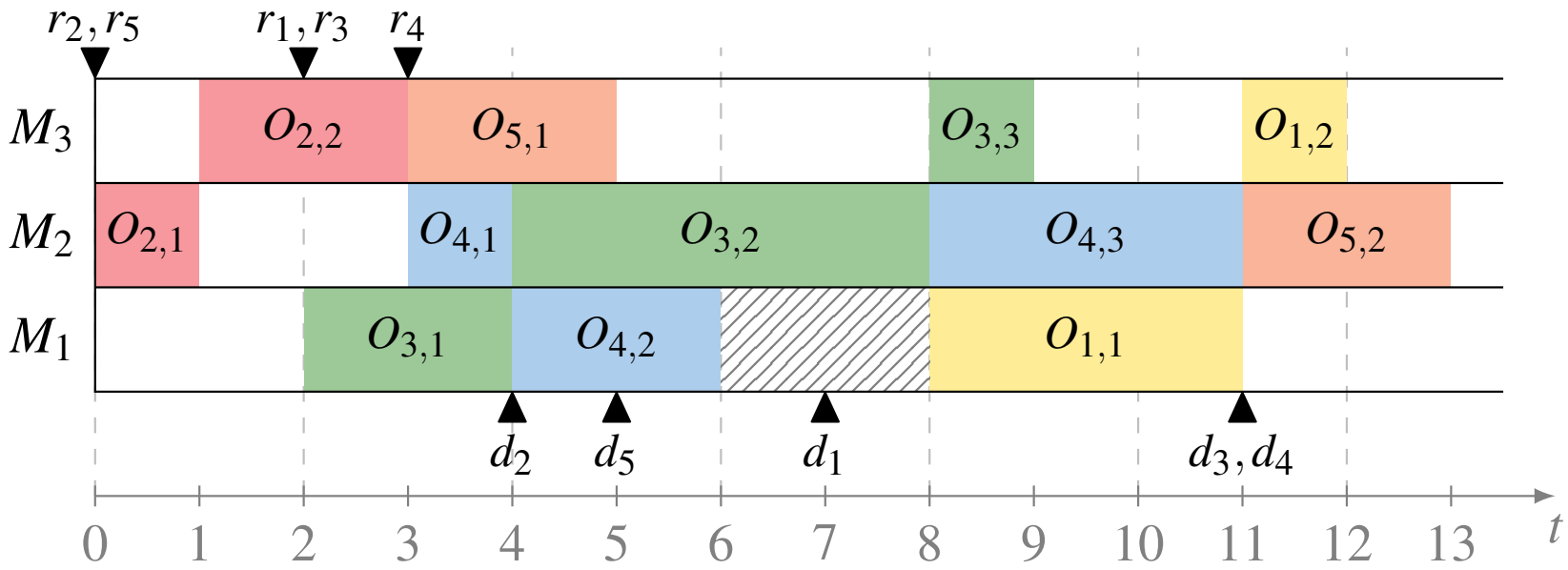
A **schedule** is to be determined by the starting times $s_{i,j}$ of all operations, so that the **total tardiness** $\sum T_i$ of all jobs is minimal.

BJSPT Instance



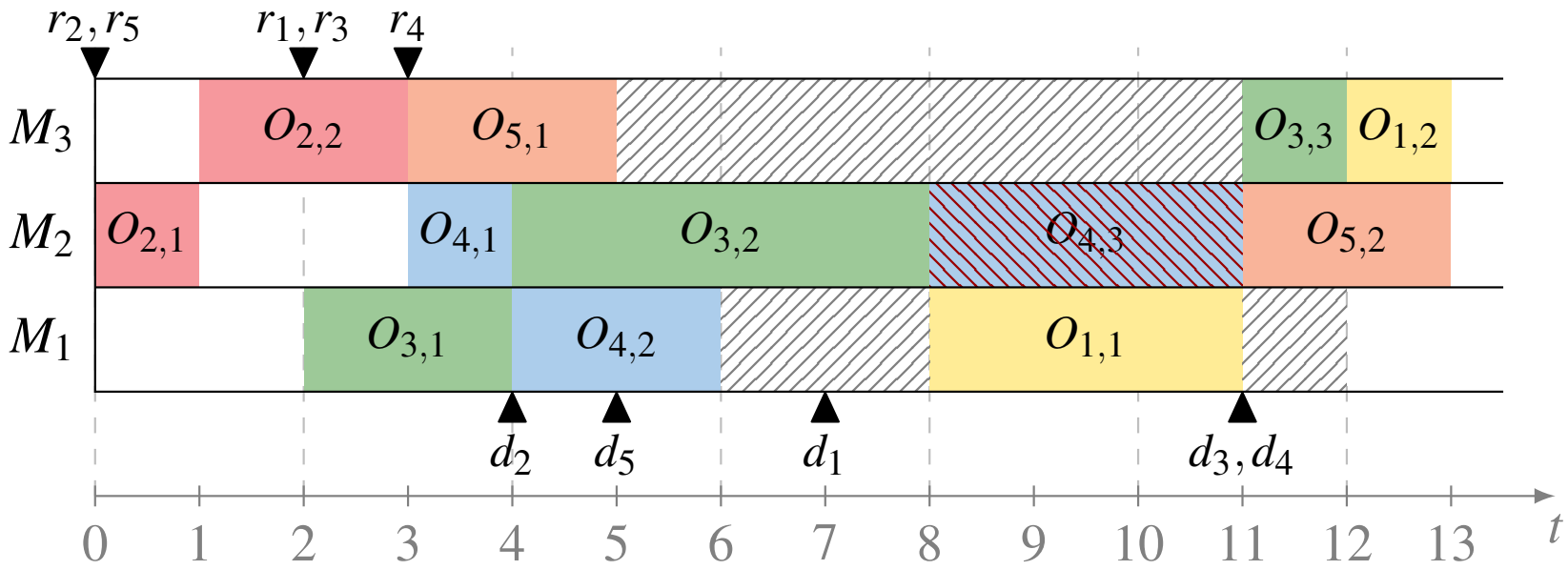
J_1	$\mathrm{TR}_1\colon M_1 \to M_3$	$p_{1,1} = 3,\ p_{1,2} = 1$
J_2	$\mathrm{TR}_2\colon M_2 \to M_3$	$p_{2,1} = 1,\ p_{2,2} = 2$
J_3	$\mathrm{TR}_3\colon M_1 \to M_2 \to M_3$	$p_{3,1} = 2,\ p_{3,2} = 4,\ p_{3,3} = 1$
J_4	$\mathrm{TR}_4\colon M_2 \to M_1 \to M_2$	$p_{4,1} = 1,\ p_{4,2} = 2,\ p_{4,3} = 3$
J_5	$\mathrm{TR}_5\colon M_3 \to M_2$	$p_{5,1} = 2,\ p_{5,2} = 2$

BJSPT Instance



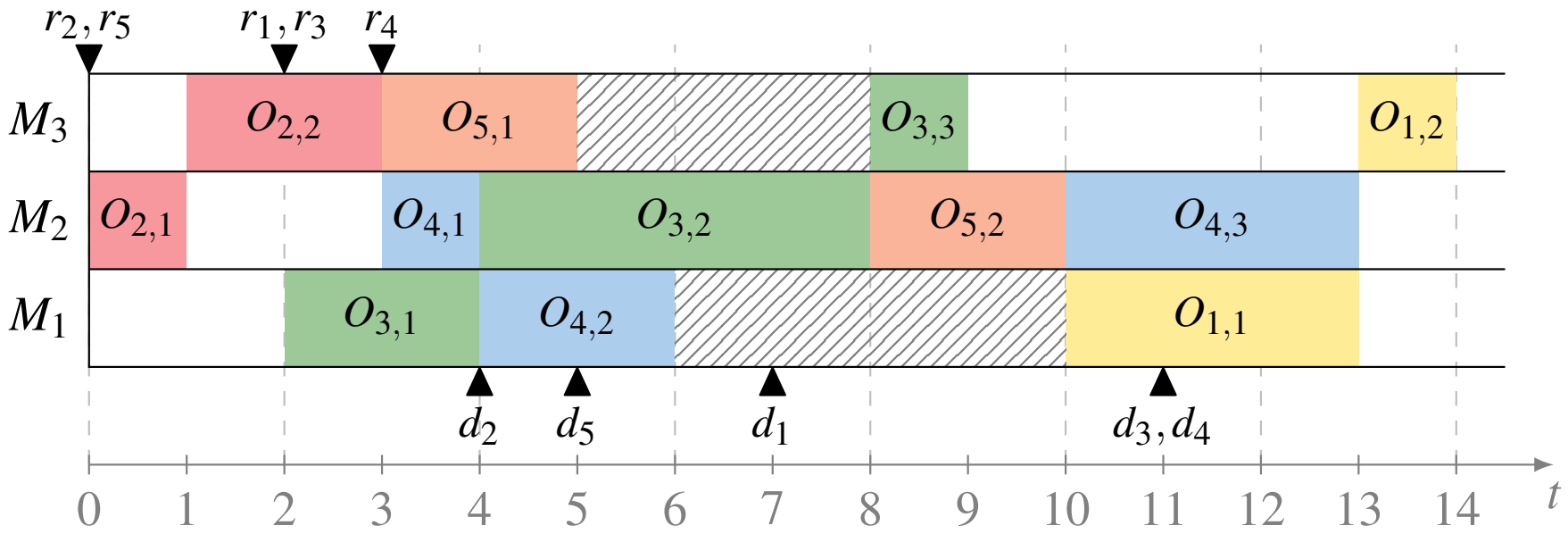
J_1	$\mathrm{TR}_1\colon M_1 \to M_3$	$p_{1,1} = 3,\ p_{1,2} = 1$
J_2	$\mathrm{TR}_2\colon M_2 \to M_3$	$p_{2,1} = 1,\ p_{2,2} = 2$
J_3	$\mathrm{TR}_3\colon M_1 \to M_2 \to M_3$	$p_{3,1} = 2,\ p_{3,2} = 4,\ p_{3,3} = 1$
J_4	$\mathrm{TR}_4\colon M_2 \to M_1 \to M_2$	$p_{4,1} = 1,\ p_{4,2} = 2,\ p_{4,3} = 3$
J_5	$\mathrm{TR}_5\colon M_3 \to M_2$	$p_{5,1} = 2,\ p_{5,2} = 2$

BJSPT Instance



J_1	TR_1: $M_1 \to M_3$	$p_{1,1} = 3,\ p_{1,2} = 1$
J_2	TR_2: $M_2 \to M_3$	$p_{2,1} = 1,\ p_{2,2} = 2$
J_3	TR_3: $M_1 \to M_2 \to M_3$	$p_{3,1} = 2,\ p_{3,2} = 4,\ p_{3,3} = 1$
J_4	TR_4: $M_2 \to M_1 \to M_2$	$p_{4,1} = 1,\ p_{4,2} = 2,\ p_{4,3} = 3$
J_5	TR_5: $M_3 \to M_2$	$p_{5,1} = 2,\ p_{5,2} = 2$

BJSPT Instance



changing the operation sequence on machine M_2

$$\sum_{J_i \in \mathcal{J}} T_i = 7 + 0 + 0 + 2 + 5 = 14$$

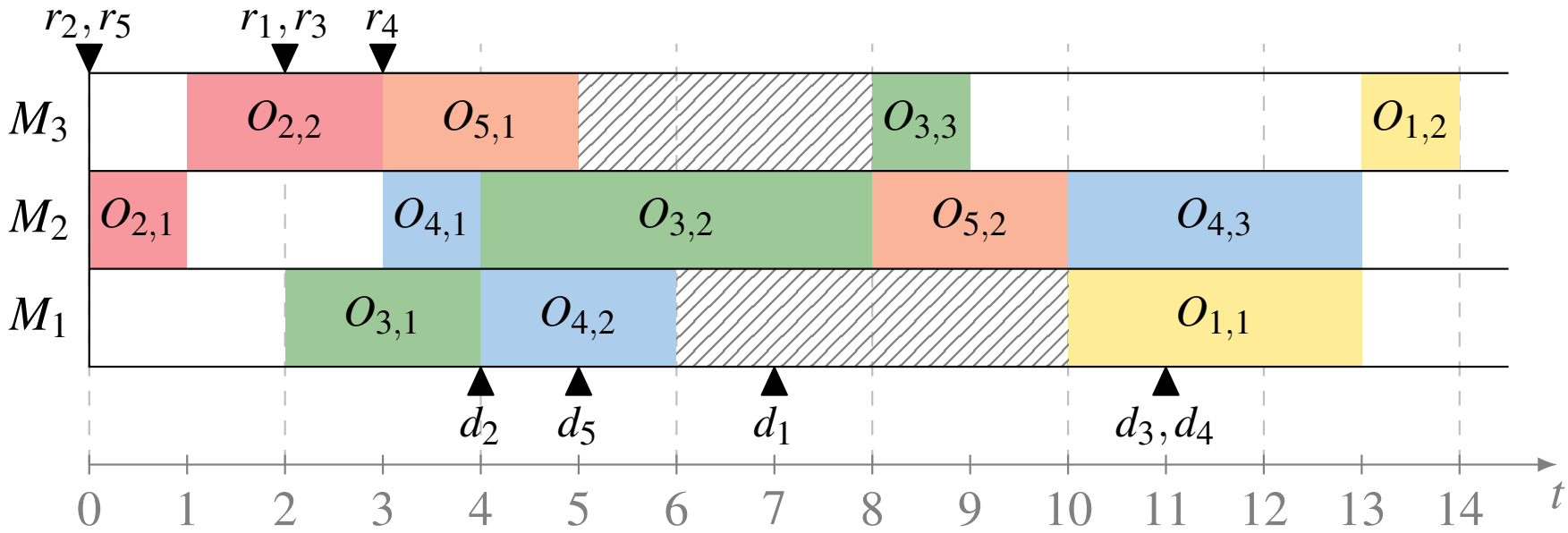
Heuristic Method

Challenge. State-of-the-art mixed-integer programming solvers struggle in finding optimal and even feasible schedules for instances with more than 150 operations.

Idea. Applying successful combinatorial search techniques from standard job shop scheduling to instances with blocking constraints and total tardiness minimization.

Implementation. Simulated Annealing algorithm with permutation-based solution representations and interchange- and shift-based neighborhoods.

Permutation-based Encodings



redundancy vs. feasibility

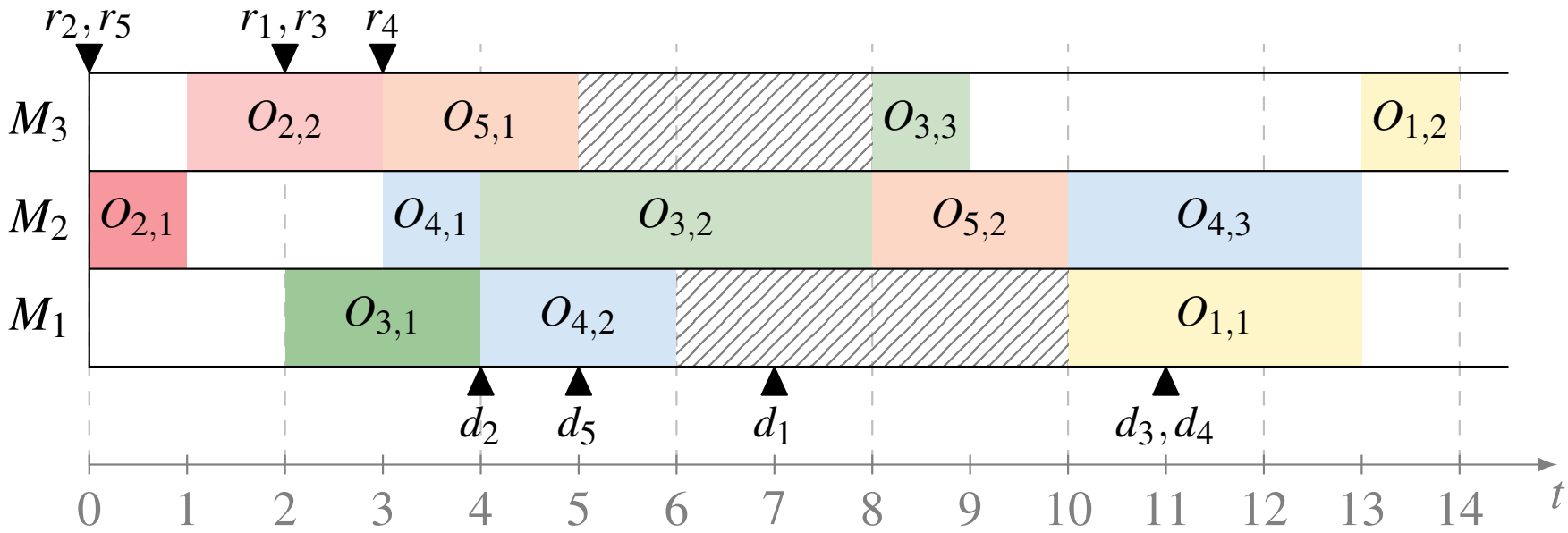
MACHINE-BASED REPRESENTATION

$$s^{ma} = [[O_{3,1}, O_{4,2}, O_{1,1}], [O_{2,1}, O_{4,1}, O_{3,2}, O_{5,2}, O_{4,3}], [O_{2,2}, O_{5,1}, O_{3,3}, O_{1,2}]]$$

OPERATION-BASED REPRESENTATION (also: permutation, list)

$$s^{op} = [O_{2,1}, O_{3,1}, O_{4,1}, O_{2,2}, O_{3,2}, O_{4,2}, O_{5,1}, O_{3,3}, O_{5,2}, O_{4,3}, O_{1,1}, O_{1,2}]$$

Permutation-based Encodings



redundancy vs. feasibility

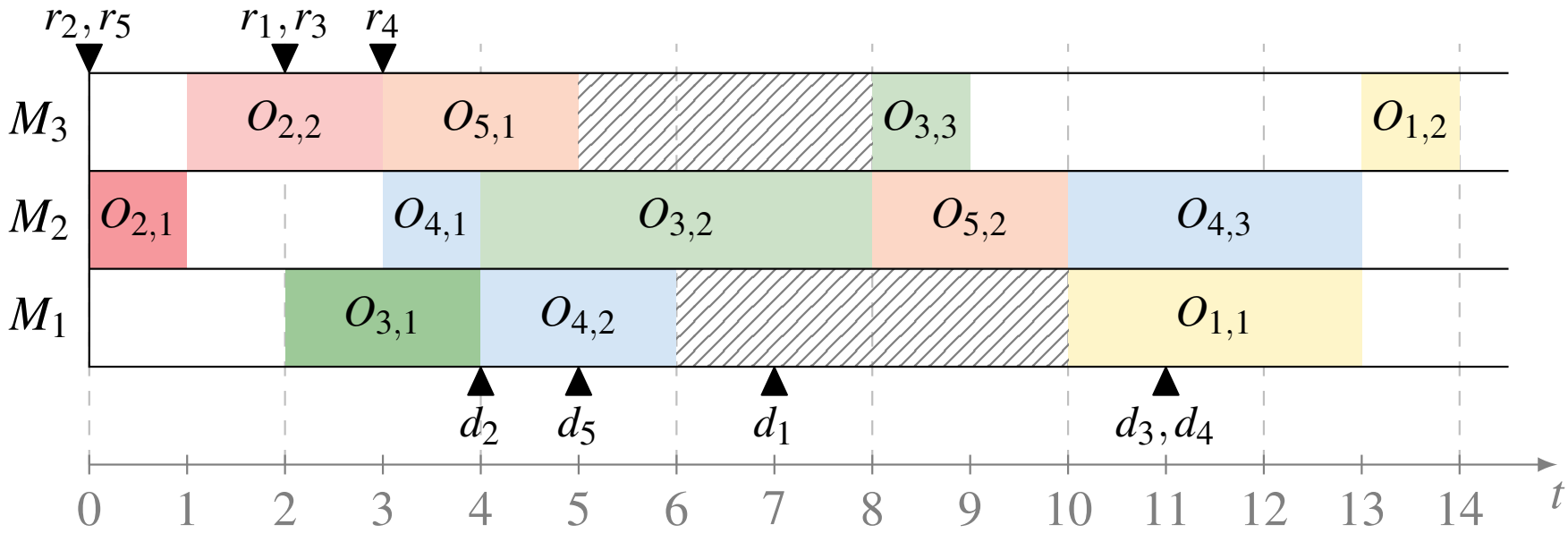
MACHINE-BASED REPRESENTATION

$$s^{ma} = [[O_{3,1}, O_{4,2}, O_{1,1}], [O_{2,1}, O_{4,1}, O_{3,2}, O_{5,2}, O_{4,3}], [O_{2,2}, O_{5,1}, O_{3,3}, O_{1,2}]]$$

OPERATION-BASED REPRESENTATION (also: permutation, list)

$$s^{op} = [O_{2,1}, O_{3,1}, O_{4,1}, O_{2,2}, O_{3,2}, O_{4,2}, O_{5,1}, O_{3,3}, O_{5,2}, O_{4,3}, O_{1,1}, O_{1,2}]$$

Permutation-based Encodings



redundancy vs. feasibility

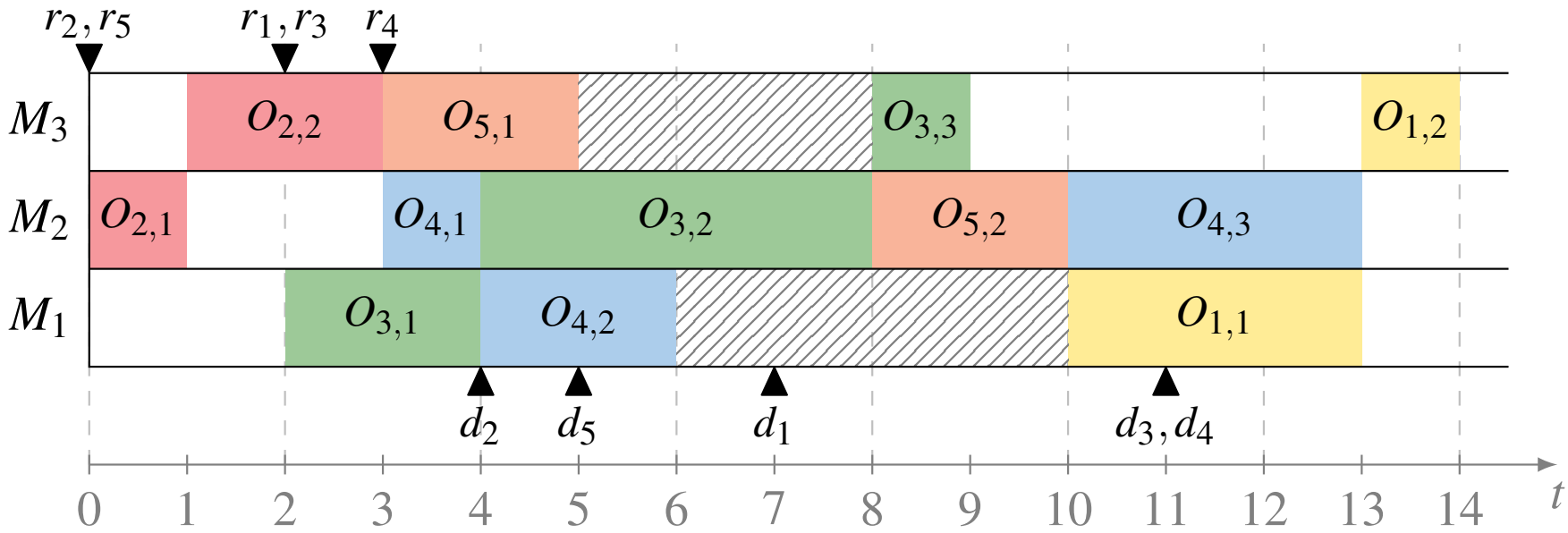
MACHINE-BASED REPRESENTATION

$$s^{ma} = [[O_{3,1}, O_{4,2}, O_{1,1}], [O_{2,1}, O_{4,1}, O_{3,2}, O_{5,2}, O_{4,3}], [O_{2,2}, O_{5,1}, O_{3,3}, O_{1,2}]]$$

OPERATION-BASED REPRESENTATION (also: permutation, list)

$$s^{op} = [O_{2,1}, O_{3,1}, O_{4,1}, O_{2,2}, O_{3,2}, O_{4,2}, O_{5,1}, O_{3,3}, O_{5,2}, O_{4,3}, O_{1,1}, O_{1,2}]$$

Permutation-based Encodings



redundancy vs. feasibility

MACHINE-BASED REPRESENTATION

$$s^{ma} = [[O_{3,1}, O_{4,2}, O_{1,1}], [O_{2,1}, O_{4,1}, O_{3,2}, O_{5,2}, O_{4,3}], [O_{2,2}, O_{5,1}, O_{3,3}, O_{1,2}]]$$

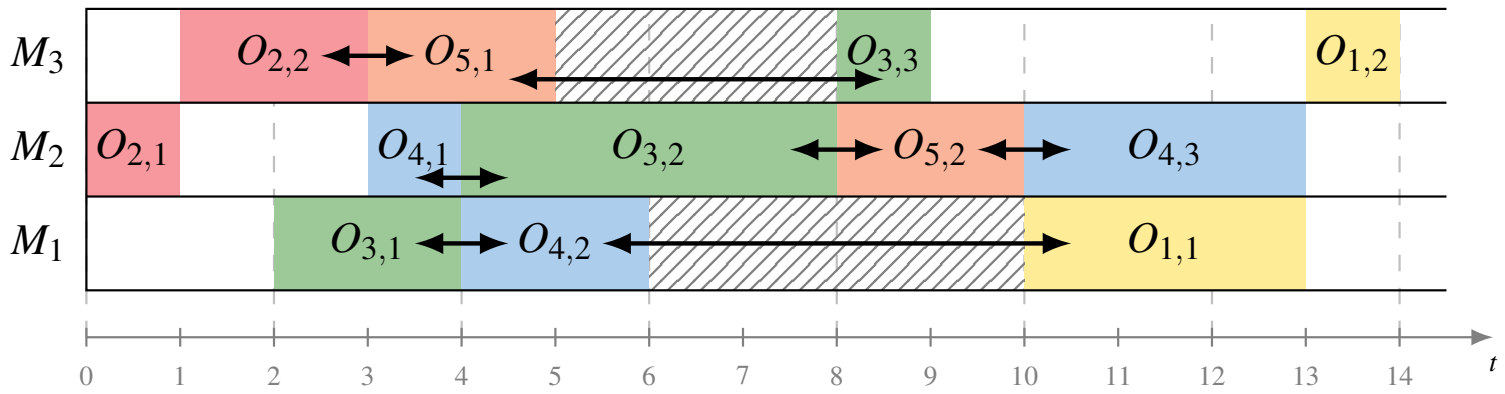
OPERATION-BASED REPRESENTATION (also: permutation, list)

$$s^{op} = [O_{2,1}, O_{3,1}, O_{2,2}, O_{4,1}, O_{3,2}, O_{4,2}, O_{5,1}, O_{3,3}, O_{5,2}, O_{4,3}, O_{1,1}, O_{1,2}]$$

← Basic Repair Technique (BRT)

API-based Neighborhood

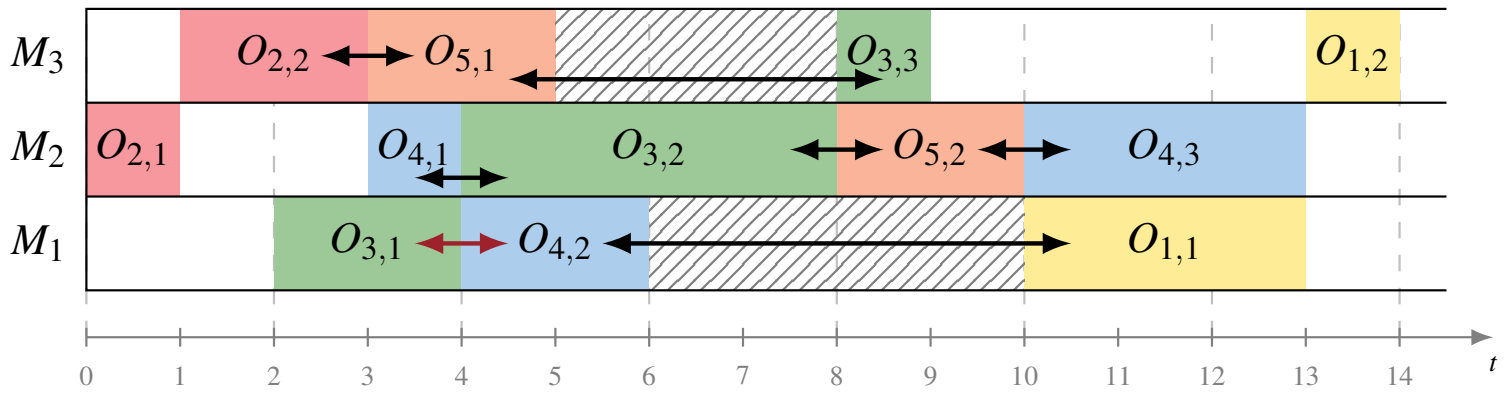
Definition 1. An *Adjacent Pairwise Interchange (API)* denotes the swap of two adjacent operations $O_{i,j}$ and $O_{i',j'}$ of different jobs on the same machine in the machine-based representation of the schedule.



$$s^{ma} = [[O_{3,1}, O_{4,2}, O_{1,1}], [O_{2,1}, O_{4,1}, O_{3,2}, O_{5,2}, O_{4,3}], [O_{2,2}, O_{5,1}, O_{3,3}, O_{1,2}]]$$

API-based Neighborhood

Definition 1. An *Adjacent Pairwise Interchange (API)* denotes the swap of two adjacent operations $O_{i,j}$ and $O_{i',j'}$ of different jobs on the same machine in the machine-based representation of the schedule.



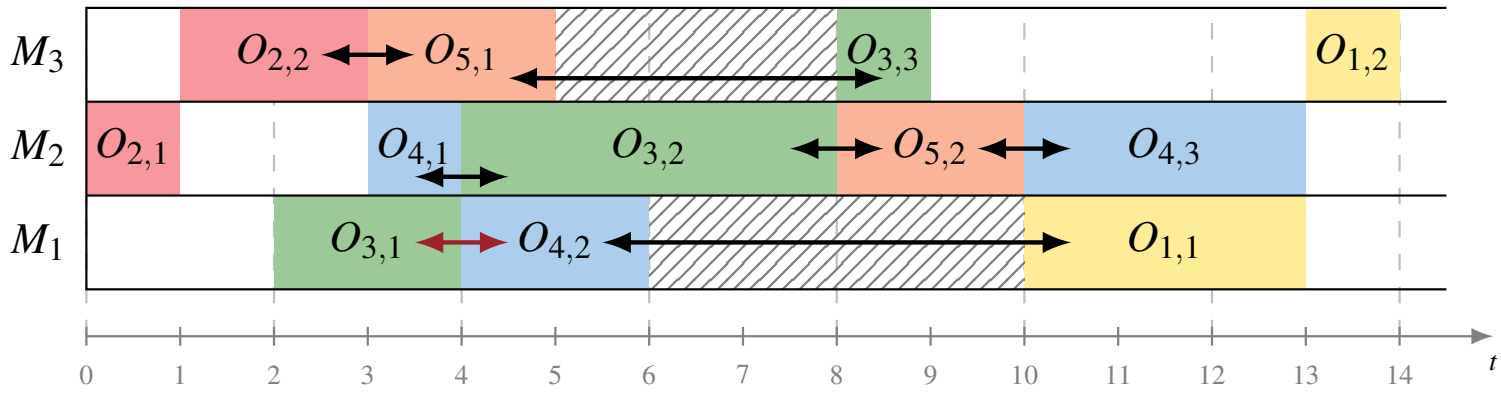
$$s^{ma} = [[O_{3,1}, O_{4,2}, O_{1,1}], [O_{2,1}, O_{4,1}, O_{3,2}, O_{5,2}, O_{4,3}], [O_{2,2}, O_{5,1}, O_{3,3}, O_{1,2}]]$$

API NEIGHBORS

$$s^{ma'} = [[\underline{O_{4,2}}, \underline{O_{3,1}}, O_{1,1}], [O_{2,1}, O_{4,1}, O_{3,2}, O_{5,2}, O_{4,3}], [O_{2,2}, O_{5,1}, O_{3,3}, O_{1,2}]]$$

API-based Neighborhood

Definition 1. An *Adjacent Pairwise Interchange (API)* denotes the swap of two adjacent operations $O_{i,j}$ and $O_{i',j'}$ of different jobs on the same machine in the machine-based representation of the schedule.



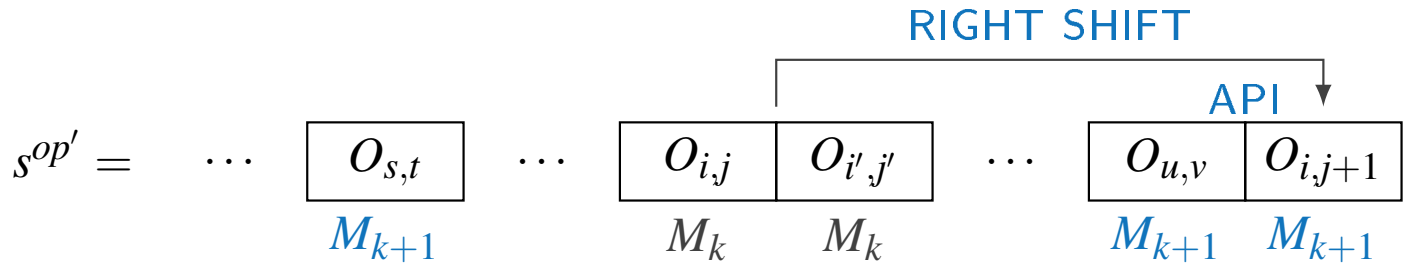
$$s^{ma} = [[O_{3,1}, O_{4,2}, O_{1,1}], [O_{2,1}, O_{4,1}, O_{3,2}, O_{5,2}, O_{4,3}], [O_{2,2}, O_{5,1}, O_{3,3}, O_{1,2}]]$$

API NEIGHBORS

$$s^{ma'} = [[\underline{O_{4,2}}, \underline{O_{3,1}}, O_{1,1}], [O_{2,1}, O_{4,1}, O_{3,2}, O_{5,2}, O_{4,3}], [O_{2,2}, O_{5,1}, O_{3,3}, O_{1,2}]]$$

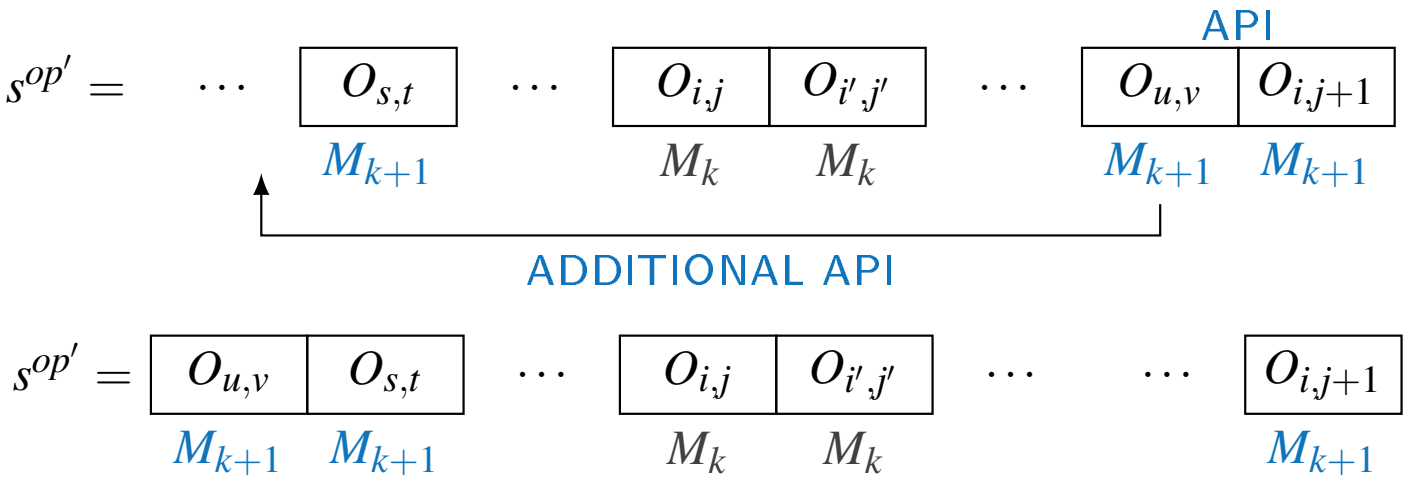
$$s^{op} = [O_{2,1}, \underline{O_{3,1}}, O_{2,2}, O_{4,1}, O_{5,1}, \underline{O_{4,2}}, O_{3,2}, O_{3,3}, O_{5,2}, O_{4,3}, O_{1,1}, O_{1,2}]$$

Challenge. The partial sequence given by API needs to be merged with the initial schedule while doing as few changes as possible.



→ Applying BRT reverts API.

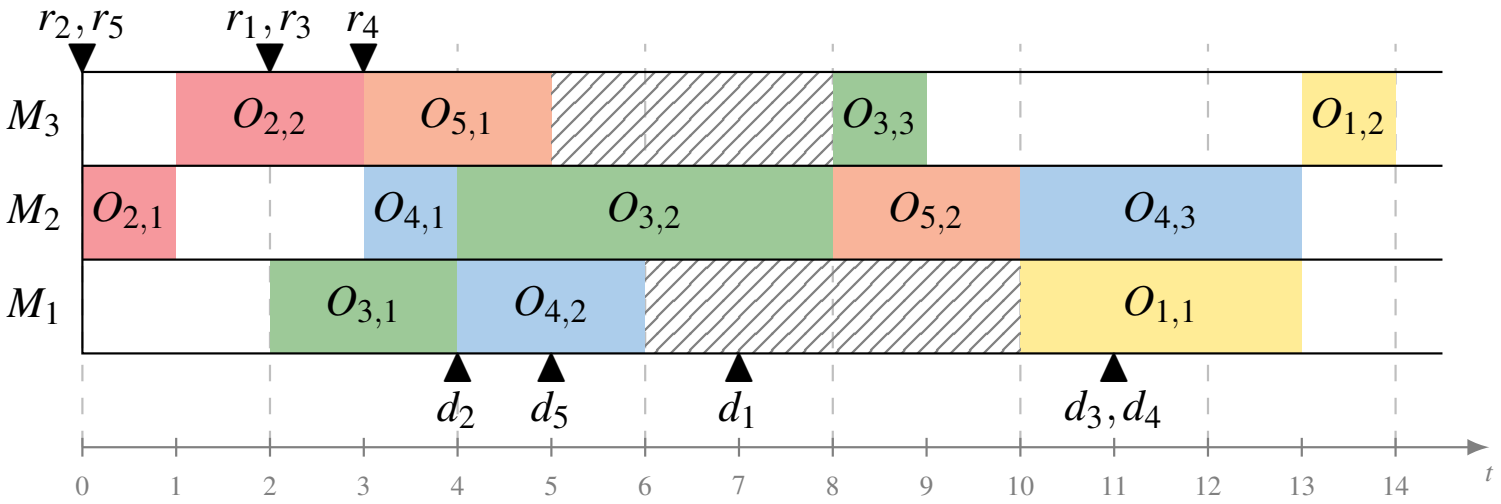
Strategy. Interchange the preceding operation (here: $O_{u,v}$) with its predecessor on its machine (here: $O_{s,t}$).



→ Applying BRT constructs feasible schedule involving partial sequence.

Randomized Neighborhood

Definition 2. A *TJ neighbor* of a schedule is determined by randomly shifting all operations $O_{i,j}$ of a randomly chosen tardy job J_i to the left in the operation-based representation of the schedule.



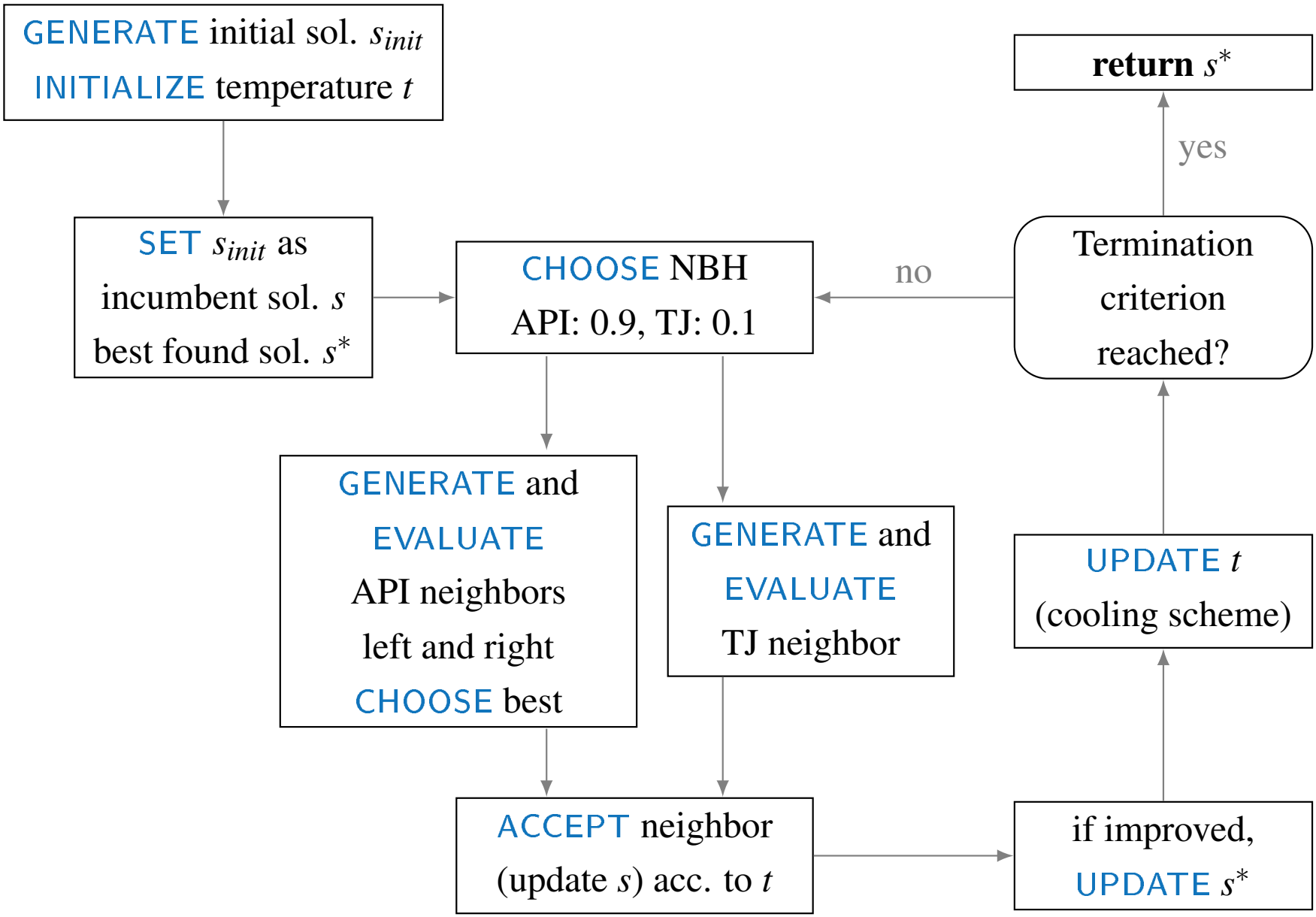
set of tardy jobs: $\{J_1, J_4, J_5\}$, randomly chosen: job J_5

$$s^{op} = [O_{2,1}, O_{3,1}, O_{2,2}, O_{4,1}, \underline{O_{5,1}}, O_{4,2}, O_{3,2}, O_{3,3}, \underline{O_{5,2}}, O_{4,3}, O_{1,1}, O_{1,2}]$$

$$s^{op'} = [O_{2,1}, O_{3,1}, \underline{O_{5,1}}, O_{2,2}, O_{4,1}, \underline{O_{5,2}}, O_{4,2}, O_{3,2}, O_{3,3}, O_{4,3}, O_{1,1}, O_{1,2}]$$

$\rightarrow$ Applying BRT.

Simulated Annealing Algorithm



Computational Results

Inst.	(m,n)	MIP	#(It.)	$\overline{\sum T_i}$	$\min\{\sum T_i\}$	$\overline{\text{Time}}$ (in s)
ts01	$(11,10)$	138*	33264	140.2	138*	384.99
ts05	$(11,10)$	71*	31878	71.4	71*	404.85
la17	$(10,10)$	118*	22160	144.2	120	341.19
la20	$(10,10)$	42*	22160	55.6	42*	330.91
ts07	$(11,15)$	172*	58905	195.4	189	880.14
ts10	$(11,15)$	97*	56133	114.8	97*	767.11
la21	$(10,15)$	1956	36010	2847.2	2101	675.21
la23	$(10,15)$	3436	36010	3692.6	3506	650.58
ts14	$(11,20)$	459	81774	462.8	418	1398.62
ts15	$(11,20)$	418	83160	419.4	401	1390.91
la27	$(10,20)$	8915	49860	7588.0	6596	1304.75
la30	$(10,20)$	6655	49860	7621.6	6775	1328.15
la32	$(10,30)$	23150	77560	21991.4	20401	4779.26
la35	$(10,30)$	none	77560	21895.0	18778	4789.15

MIP results: CPLEX 12.8, runtime: 2 h, * = optimal solution

Computational Results

Inst.	(m,n)	MIP	#(It.)	$\overline{\sum T_i}$	$\min\{\sum T_i\}$	$\overline{\text{Time}}$ (in s)
ts01	$(11,10)$	138*	33264	140.2	138*	384.99
ts05	$(11,10)$	71*	31878	71.4	71*	404.85
la17	$(10,10)$	118*	22160	144.2	120	341.19
la20	$(10,10)$	42*	22160	55.6	42*	330.91
ts07	$(11,15)$	172*	58905	195.4	189	880.14
ts10	$(11,15)$	97*	56133	114.8	97*	767.11
la21	$(10,15)$	1956	36010	2847.2	2101	675.21
la23	$(10,15)$	3436	36010	3692.6	3506	650.58
ts14	$(11,20)$	459	81774	462.8	418	1398.62
ts15	$(11,20)$	418	83160	419.4	401	1390.91
la27	$(10,20)$	8915	49860	7588.0	6596	1304.75
la30	$(10,20)$	6655	49860	7621.6	6775	1328.15
la32	$(10,30)$	23150	77560	21991.4	20401	4779.26
la35	$(10,30)$	none	77560	21895.0	18778	4789.15

MIP results: CPLEX 12.8, runtime: 2 h, * = optimal solution

Computational Results

Inst.	(m,n)	MIP	#(It.)	$\overline{\sum T_i}$	$\min\{\sum T_i\}$	$\overline{\text{Time}}$ (in s)
ts01	$(11,10)$	138*	33264	140.2	138*	384.99
ts05	$(11,10)$	71*	31878	71.4	71*	404.85
la17	$(10,10)$	118*	22160	144.2	120	341.19
la20	$(10,10)$	42*	22160	55.6	42*	330.91
ts07	$(11,15)$	172*	58905	195.4	189	880.14
ts10	$(11,15)$	97*	56133	114.8	97*	767.11
la21	$(10,15)$	1956	36010	2847.2	2101	675.21
la23	$(10,15)$	3436	36010	3692.6	3506	650.58
ts14	$(11,20)$	459	81774	462.8	418	1398.62
ts15	$(11,20)$	418	83160	419.4	401	1390.91
la27	$(10,20)$	8915	49860	7588.0	6596	1304.75
la30	$(10,20)$	6655	49860	7621.6	6775	1328.15
la32	$(10,30)$	23150	77560	21991.4	20401	4779.26
la35	$(10,30)$	none	77560	21895.0	18778	4789.15

MIP results: CPLEX 12.8, runtime: 2 h, * = optimal solution

Concluding Remarks

HEURISTIC METHOD

- Good quality schedules for large problem instances
- Significantly rugged search space to explore
- Considerable amount of computation time required by complex repair mechanisms

FUTURE RESEARCH

Combination of mixed-integer programming techniques (quickly finding optimal solutions for small instances) and heuristic search (guaranteed feasible solutions)