Branch and Price for Job-Shop Scheduling with Time-Dependent Costs and Resource Constraints

Marouane Felloussi[1,2], Mohammed Ghannam[2], João Dionísio[2,3],
Paolo Gianessi[1], Xavier Delorme[1]

[1]Mines Saint-Etienne, LIMOS CNRS, Saint-Etienne, France

[2]Zuse Institute Berlin, Germany

[3]Faculdade de Ciências, Universidade do Porto, Porto, Portugal

April 14, 2026

PMS 2026, Toulouse, France

Scheduling under Time-Varying Costs and Limits

- **Scheduling** problems: **manufacturing**, workforce management, computing.
- Optimizing time-based metrics alone may lead to high **operating costs**.
- The **cost** of executing a task depends on **when** it runs:
  - electricity tariffs,
  - compute pricing.
- Resources consumed over time may be **limited**:
  - contractual capacity agreements,
  - shared power budgets.

Problem Definition

- **Machines** $m \in \mathcal{M}$.
- **Jobs** $j \in \mathcal{J}$ to execute over a discrete time horizon $\mathcal{T}$:
  - » Processed over an ordered subset of machines (each exactly once).
- **Operations** $(j, m) \in \mathcal{J} \times \mathcal{M}$: processing of job j on machine m.
  - » Deterministic processing times $q_{j,m}$.
  - » Direct precedence constraints: $(j, m) \prec (j, m')$.
- **Cost** $g^t_{j,m} \in \mathbb{R}_+$: incurred from executing (j, m) at $t \in \mathcal{T}$.
- **Cardinality resource limit** $\overline{M}_t$: maximum simultaneously active machines at t.

Problem Definition

A **feasible solution** is a **schedule** where each operation (j, m) executes such that:

- same-machine operations do not process simultaneously **(non-overlap)**,
- operations may not interrupt processing **(non-preemption)**,
- operation sequencing respects a predefined order **(precedence)**,
- machine usage at any time does not exceed the limit **(cardinality constraint)**.

Objective

Find a **feasible** solution **minimizing** the Total time-dependent Operating Cost (TOC).

Energy-aware scheduling: instance and solution

Job j	1	2	3
Sequence M_j	$\{1,3,2\}$	$\{2,1,3\}$	$\{2,3,1\}$

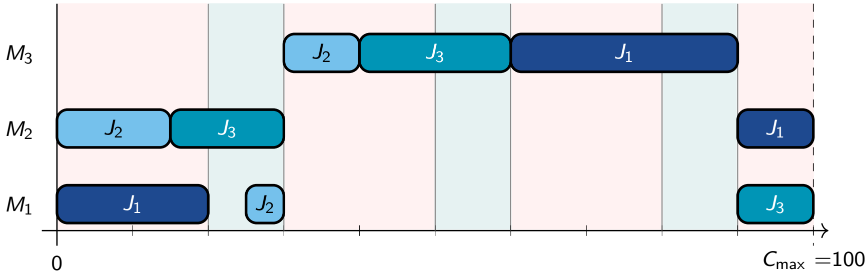
Cost period p	on-peak	off-peak
Tariff c^p	0.159	0.13

Machine m	1	2	3
duration $q_{1,m}$	20	10	30
duration $q_{2,m}$	5	15	10
duration $q_{3,m}$	10	15	20

Machine m	1	2	3
Power φ_m	5	6	8

$$g_{j,m}^t = \varphi_m \times c^p, \forall t \in p$$

Makespan minimization s.t. $\overline{M} = 2$ (TOC $= 204.7$)



Energy-aware scheduling: instance and solution

Job j	1	2	3
Sequence M_j	$\{1,3,2\}$	$\{2,1,3\}$	$\{2,3,1\}$

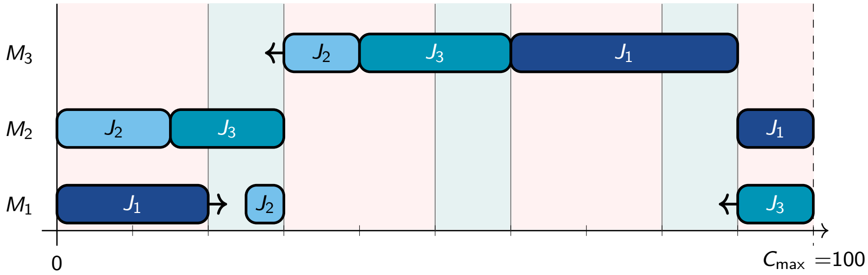
Cost period p	on-peak	off-peak
Tariff c^p	0.159	0.13

Machine m	1	2	3
duration $q_{1,m}$	20	10	30
duration $q_{2,m}$	5	15	10
duration $q_{3,m}$	10	15	20

Machine m	1	2	3
Power φ_m	5	6	8

$$g_{j,m}^t = \varphi_m \times c^p, \forall t \in p$$

Makespan minimization s.t. $\overline{M} = 2$ (TOC $= 204.7$)



Energy-aware scheduling: instance and solution

Job j	1	2	3
Sequence M_j	$\{1,3,2\}$	$\{2,1,3\}$	$\{2,3,1\}$

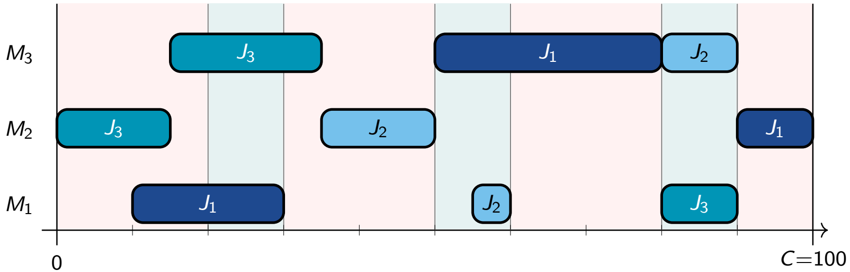
Cost period p	on-peak	off-peak
Tariff c^p	0.159	0.13

Machine m	1	2	3
duration $q_{1,m}$	20	10	30
duration $q_{2,m}$	5	15	10
duration $q_{3,m}$	10	15	20

Machine m	1	2	3
Power φ_m	5	6	8

$$g_{j,m}^t = \varphi_m \times c^p, \forall t \in p$$

Operating Cost minimization s.t. $\overline{M} = 2$ and $C_{\max}$ (TOC $= 197.1$)



Short literature review: energy-aware scheduling

Problem class	Article	Problem	Solution Approach
job-shop scheduling	Felloussi et al. [2026]	$Jm \vert P_{max} \vert TOC$	MILP (PI), B&C
	Felloussi et al. [2025]	$Jm \vert\vert TOC$	MILP (PI)
	Bley and Linß [2022]	$Jm \vert on/off, r_j, d_j \vert TOC$	MILP (TI), B&B
	Masmoudi et al. [2019]	$Jm \vert P_{max} \vert TOC$	MILP (TI), MH
flexible job-shop scheduling	Park and Ham [2022]	$FJm \vert on/off \vert C_{max}, TOC$	MILP (TI), CP
	Jiang and Wang [2020]	$FJm \vert\vert C_{max}, TOC$	MILP (TI), H
flow-shop scheduling	Ho et al. [2022]	$F2 \vert prmu, on/off \vert TOC$	MILP (PI), B&C
parallel machine scheduling	Gaggero et al. [2023]	$Pm \vert\vert C_{max}, TOC$	MILP (TI), H
	Che et al. [2017]	$Rm \vert\vert TOC$	MILP (PI), H
	Ding et al. [2016]	$Rm \vert\vert TOC$	MILP (PI), CG-H
single machine scheduling	Tian and Zheng [2024]	$1 \vert batch \vert TOC$	MILP (PI), CG-H
	Cheng et al. [2016]	$1 \vert batch \vert TOC$	MILP (PI)

PI: Period-Indexed formulation   TI: Time-Indexed formulation

Table: An overview of related works.

- Piecewise-constant cost profiles ⇒ both PI and TI are applicable.
- In practice: #periods ≪ #time steps ⇒ PI outperform TI [Felloussi et al., 2026].

Short literature review: energy-aware scheduling

Problem class	Article	Problem	Solution Approach
job-shop scheduling	Felloussi et al. [2026]	$Jm\|\mathrm{P}_{\max}\|\mathrm{TOC}$	MILP (PI), B&C
	Felloussi et al. [2025]	$Jm\|\|\mathrm{TOC}$	MILP (PI)
	Bley and Linß [2022]	$Jm\|\,\mathrm{on/off}, r_j, d_j\|\mathrm{TOC}$	MILP (TI), B&B
	Masmoudi et al. [2019]	$Jm\|\mathrm{P}_{\max}\|\mathrm{TOC}$	MILP (TI), MH
flexible job-shop scheduling	Park and Ham [2022]	$FJm\|\,\mathrm{on/off}\,\|C_{\max},\mathrm{TOC}$	MILP (TI), CP
	Jiang and Wang [2020]	$FJm\|\|C_{\max},\mathrm{TOC}$	MILP (TI), H
flow-shop scheduling	Ho et al. [2022]	$F2\|prmu,\mathrm{on/off}\,\|\mathrm{TOC}$	MILP (PI), B&C
parallel machine scheduling	Gaggero et al. [2023]	$Pm\|\|C_{\max},\mathrm{TOC}$	MILP (TI), H
	Che et al. [2017]	$Rm\|\|\mathrm{TOC}$	MILP (PI), H
	Ding et al. [2016]	$Rm\|\|\mathrm{TOC}$	MILP (PI), CG-H
single machine scheduling	Tian and Zheng [2024]	$1\|batch\|\mathrm{TOC}$	MILP (PI), CG-H
	Cheng et al. [2016]	$1\|batch\|\mathrm{TOC}$	MILP (PI)

PI: Period-Indexed formulation TI: Time-Indexed formulation

Table: An overview of related works.

- Piecewise-constant cost profiles ⇒ both PI and TI are applicable.
- In practice: #periods ≪ #time steps ⇒ PI outperform TI [Felloussi et al., 2026].

→ What if the costs (or resource limits) vary at **unit-time resolution**?

Cost and limit variation at unit-time resolution

- Practically relevant in many settings:
  - » Some electricity markets price at dispatch frequency (5-min, U.S. EIA [2022]).
  - » Intermittent renewables induce price volatility (15-min, Nord Pool [2025]).
  - » Shared power capacity: non-controllable loads induce time-varying resource limits.

- Unit-time resolution $\Rightarrow$ TI models are natural but too large to solve directly.

Cost and limit variation at unit-time resolution

- Practically relevant in many settings:
  - » Some electricity markets price at dispatch frequency (5-min, U.S. EIA [2022]).
  - » Intermittent renewables induce price volatility (15-min, Nord Pool [2025]).
  - » Shared power capacity: non-controllable loads induce time-varying resource limits.
- Unit-time resolution $\Rightarrow$ TI models are natural but too large to solve directly.

This work

Manage the model size via an **extended formulation** and solve by **Branch and Price**.

Outline

Time-Indexed Formulations

- Binary variables indicate whether (j, m) is "processed" at step t.
- Two compact variants [Artigues, 2017]:
  - » **Disaggregated**: strong LP relaxation, large model.
  - » **Aggregated**: weaker LP relaxation, smaller model.
- **Main drawback**: size pseudo-polynomial in $|\mathcal{T}|$.

Time-Indexed Formulations

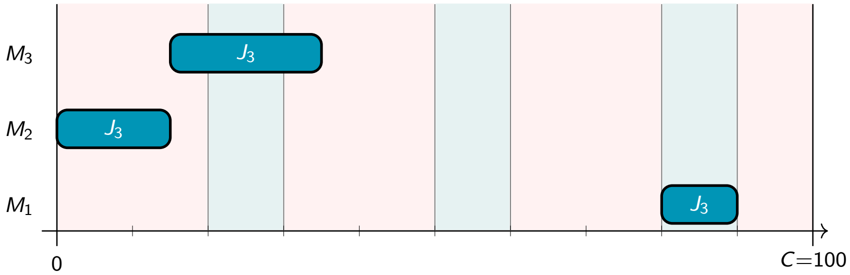
- Binary variables indicate whether (j, m) is "processed" at step t.
- Two compact variants [Artigues, 2017]:
  - » **Disaggregated**: strong LP relaxation, large model.
  - » **Aggregated**: weaker LP relaxation, smaller model.
- **Main drawback**: size pseudo-polynomial in $|\mathcal{T}|$.

Idea

Decompose into **job-schedules**: offload part of the combinatorial work to a fast oracle.

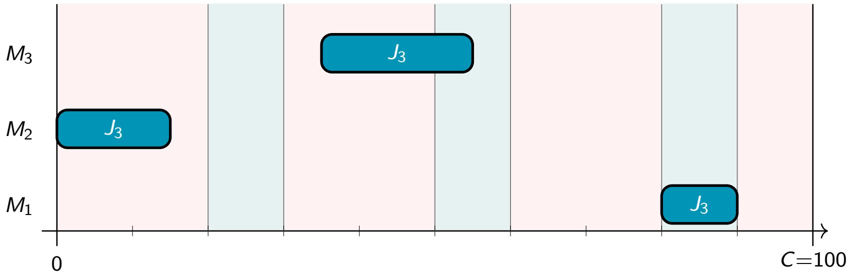
Job-schedules

- Full schedule is decomposed into $|\mathcal{J}|$ **job-schedules**.
- For job j, a job-schedule $p \in \mathcal{P}_j$ is a feasible pseudo-schedule where operations:
  - » fully execute non-preemptively,
  - » follow their predefined precedence sequence.
- Binary profile $a^p_{m,t} \in \{0, 1\} \equiv$ process on m at t, total cost $c_p := \sum\limits_{m \in \mathcal{M}} \sum\limits_{t \in \mathcal{T}} g^t_{j,m} a^p_{m,t}$.



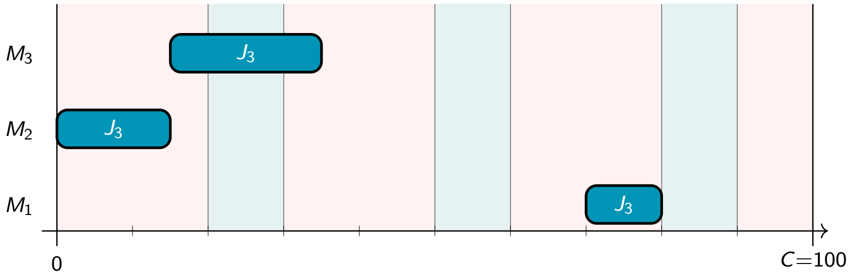
Job-schedules

- Full schedule is decomposed into $|\mathcal{J}|$ **job-schedules**.
- For job j, a job-schedule $p \in \mathcal{P}_j$ is a feasible pseudo-schedule where operations:
  - » fully execute non-preemptively,
  - » follow their predefined precedence sequence.
- Binary profile $a^p_{m,t} \in \{0,1\} \equiv$ process on m at t, total cost $c_p := \sum_{m \in \mathcal{M}} \sum_{t \in \mathcal{T}} g^t_{j,m} a^p_{m,t}$.



Job-schedules

- Full schedule is decomposed into $|\mathcal{J}|$ **job-schedules**.
- For job j, a job-schedule $p \in \mathcal{P}_j$ is a feasible pseudo-schedule where operations:
  - » fully execute non-preemptively,
  - » follow their predefined precedence sequence.
- Binary profile $a^p_{m,t} \in \{0,1\} \equiv$ process on m at t, total cost $c_p := \sum_{m\in\mathcal{M}} \sum_{t\in\mathcal{T}} g^t_{j,m} a^p_{m,t}$.



Extended Formulation

$$
\textbf{(MP)} \quad \begin{aligned}
\min \quad & \sum_{j\in\mathcal{J}} \sum_{p\in\mathcal{P}_j} c_p \lambda_p, & & \\
\text{s.t.} \quad & \sum_{p\in\mathcal{P}_j} \lambda_p = 1, & \forall j \in \mathcal{J}, \quad & \text{(job assignment)} \\
& \sum_{j\in\mathcal{J}} \sum_{p\in\mathcal{P}_j} a^p_{m,t} \lambda_p \leqslant 1, & \forall m \in \mathcal{M}, \forall t \in \mathcal{T}, \quad & \text{(disjunction)} \\
& \sum_{j\in\mathcal{J}} \sum_{p\in\mathcal{P}_j} \sum_{m\in\mathcal{M}} a^p_{m,t} \lambda_p \leqslant \overline{M}_t, & \forall t \in \mathcal{T}, \quad & \text{(cardinality)} \\
& \lambda^p \in \{0,1\} & \forall p \in \mathcal{P} := \bigcup_{j\in\mathcal{J}} \mathcal{P}_j. &
\end{aligned} \tag{1}
$$

- $|\mathcal{P}|$ is exponentially large $\Rightarrow$ manage via **column generation**.
- LP relaxation solved at each node of a branch-and-bound tree $\Rightarrow$ **Branch and Price**.

Core algorithm

- Relax $\lambda^p \in \{0,1\}$ to $\lambda^p \geqslant 0$ and **restrict (MP)** to $\mathcal{P}' \subseteq \mathcal{P} \Rightarrow$ **(RMP)**.
- **Column generation** iterates:
  1. Solve **(RMP)** $\rightarrow$ optimal duals $(\hat{\alpha}, \hat{\beta}, \hat{\gamma})$.
  2. Solve **pricing subproblem**: find p with most negative reduced cost,

  $$\bar{c}_p = \min_p \{ c_p - \hat{\alpha}_j - \textstyle\sum_{m,t} \hat{\beta}_{m,t} a^p_{m,t} - \sum_t \hat{\gamma}_t \sum_m a^p_{m,t} \}.$$

  3. If $\bar{c}_p$: add p to $\mathcal{P}'$, re-optimize **(RMP)**; else: **(MP)** is solved.
- **Branching** restores integrality.

Outline

Pricing Subproblem

- The pricing problem is a **shortest-path** on a directed acyclic graph (DAG):
  - » Nodes: states $(i, t) \equiv$ operation i completing at time t.
  - » *Idle* arcs: $(i, t) \rightarrow (i, t+1)$ with length zero.
  - » *Processing* arcs: $(i-1, t-q_i) \rightarrow (i, t)$ with length $=$ window sums of reduced costs.
- Solved **efficiently** by a **forward dynamic program**.

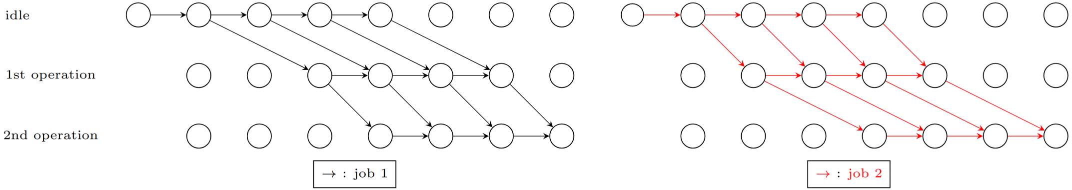


Figure: DAGs for pricing ($|\mathcal{J}| = |\mathcal{M}| = 2$)

LP Bound and Computational Advantage

- Decompositions over **sequenced activities** have been used in **precedence-constrained** settings [Drexl and Kimms, 2001, Lancia et al., 2011].
- The induced path polytope is **integral** $\Rightarrow$ the extended and disaggregated formulations share the **same LP bound**.
- The integrality property, dual oscillations, and degeneracy: **known challenges** in B&P for scheduling [Kolter et al., 2024].

LP Bound and Computational Advantage

- Decompositions over **sequenced activities** have been used in **precedence-constrained** settings [Drexl and Kimms, 2001, Lancia et al., 2011].
- The induced path polytope is **integral** $\Rightarrow$ the extended and disaggregated formulations share the **same LP bound**.
- The integrality property, dual oscillations, and degeneracy: **known challenges** in B&P for scheduling [Kolter et al., 2024].

In this setting

Disaggregated relaxations are already tight, decomposition allows:

» Reduced explicit model size.

» Fast pricing oracle $\Rightarrow$ repeated calls are inexpensive.

Outline

Branching and Propagation

- **Branch** in the ISP on $z_{j,m}^{t} \in \{0,1\}$: whether (j,m) processes at t.
  - Both branches reduce to arc fixings in the pricing DP.
  - Outperformed master branching and branching on starting times.
- Hierarchical variable selection:
  - most-dispersed rule $\rightarrow$ strong branching $\rightarrow$ pseudo-cost branching.

Branching and Propagation

- **Branch** in the ISP on $z^t_{j,m} \in \{0,1\}$: whether (j,m) processes at t.
  - Both branches reduce to arc fixings in the pricing DP.
  - Outperformed master branching and branching on starting times.
- Hierarchical variable selection:
  - most-dispersed rule $\to$ strong branching $\to$ pseudo-cost branching.
- **Propagation**: each branching decision triggers additional fixings from disjunction, precedence, non-preemption, and cardinality constraints.
  - Number of induced fixings $\Rightarrow$ **inference score**, combined with pseudo-costs.
  - Also invoked during strong branching.

Outline

Primal Heuristics: Motivation

- Generic MIP heuristics:
  - **Combine** columns from the **current pool** into primal solutions.
  - But, are agnostic to the CG process: **heuristic limitations** or **integer infeasible**?
- Restricted master heuristic: solve (RMP) as a 0-1 MIP
  - If feasible: finds current incumbent; else: need to add more columns.
  - Computationally challenging even in decision form.

Primal Heuristics: Motivation

- Generic MIP heuristics:
  - **Combine** columns from the **current pool** into primal solutions.
  - But, are agnostic to the CG process: **heuristic limitations** or **integer infeasible**?
- Restricted master heuristic: solve (RMP) as a 0-1 MIP
  - If feasible: finds current incumbent; else: need to add more columns.
  - Computationally challenging even in decision form.

Idea

Problem-specific **exact tree search** over the column pool with LP-free bounds, complemented with **Large Neighborhood Search (LNS)**.

Primal Heuristics

- 0-1 (RMP) over the current column pool: large but highly constrained.
- **Exact tree search**:
  - DFS where node $\equiv$ assign one column per job (set-partitioning).
  - Traversal follows order of jobs in $\downarrow$#cols and columns in $\uparrow$cost.
  - Pruning by partial cost or LP-free additive bound.

Primal Heuristics

- 0-1 (RMP) over the current column pool: large but highly constrained.
- **Exact tree search**:
  - DFS where node $\equiv$ assign one column per job (set-partitioning).
  - Traversal follows order of jobs in $\downarrow$#cols and columns in $\uparrow$cost.
  - Pruning by partial cost or LP-free additive bound.
- No feasible solution in the pool $\Rightarrow$ **LNS**: destroy-repair cycles over a neighborhood of the incumbent, seeded by partial solutions from the tree search.
- Destroy-repair calls the **same DP pricing oracle** $\Rightarrow$ **inexpensive** cycles, only add **columns** that combine into **new incumbent**.

Outline

Additional components

- **Pricing DP**
  - tightened time windows from precedence $\Rightarrow$ arc fixings,
  - window (processing arc) cost sums can be updated in $O(1)$.

Additional components

- **Pricing DP**
  - » tightened time windows from precedence $\Rightarrow$ arc fixings,
  - » window (processing arc) cost sums can be updated in $O(1)$.
- **Dual stabilization**
  - » exact pricing $\Rightarrow$ parameter-free dual **smoothing** [Pessoa et al., 2018],
  - » **dual-optimal inequality** on the set-partitioning duals.

Additional components

- **Pricing DP**
  - tightened time windows from precedence $\Rightarrow$ arc fixings,
  - window (processing arc) cost sums can be updated in $O(1)$.
- **Dual stabilization**
  - exact pricing $\Rightarrow$ parameter-free dual **smoothing** [Pessoa et al., 2018],
  - **dual-optimal inequality** on the set-partitioning duals.
- **Probability complementary pricing**: helps in finding early feasible solutions.

Additional components

- **Pricing DP**
  - » tightened time windows from precedence $\Rightarrow$ arc fixings,
  - » window (processing arc) cost sums can be updated in $O(1)$.
- **Dual stabilization**
  - » exact pricing $\Rightarrow$ parameter-free dual **smoothing** [Pessoa et al., 2018],
  - » **dual-optimal inequality** on the set-partitioning duals.
- **Probability complementary pricing**: helps in finding early feasible solutions.
- **Early branching** [Mehrotra and Trick, 1996]:
  - » exact pricing $\Rightarrow$ **Lagrangian lower bound** LB_{lag} [Lübbecke, 2011],
  - » **branch** before CG convergence if

$$\lceil z_{\text{RMP}} \rceil = \lceil LB_{\text{lag}} \rceil.$$

Outline

Instances and Setup

Instances generated following Felloussi et al. [2026]:

- 7 base JSSP instances of varying size.
- Time-dependent costs: 2 piecewise-constant profiles + 1 randomly drawn profile.
- Cardinality limit $\overline{M}_t \in \{|\mathcal{M}| - 1, |\mathcal{M}| - 2\}$.
- Horizons $|\mathcal{T}| = \lceil \omega C^\star \rceil$:
  - » $\omega \in \{1.2, 1.6, 2.0, 3.0\}$,
  - » $C^\star \in [56, 850]$.

$\Rightarrow$ **504 instances** across small, medium, and large horizon regimes.

Instances and Setup

Instances generated following Felloussi et al. [2026]:

- 7 base JSSP instances of varying size.
- Time-dependent costs: 2 piecewise-constant profiles + 1 randomly drawn profile.
- Cardinality limit $\overline{M}_t \in \{|\mathcal{M}| - 1, |\mathcal{M}| - 2\}$.
- Horizons $|\mathcal{T}| = \lceil \omega C^\star \rceil$:
  - » $\omega \in \{1.2, 1.6, 2.0, 3.0\}$,
  - » $C^\star \in [56, 850]$.

⇒ **504 instances** across small, medium, and large horizon regimes.

Setup: `SCIP` + `PySCIPOpt`, Intel Xeon Gold 5122, 1 thread, 1h TL.

Instances and Setup

Instances generated following Felloussi et al. [2026]:

- 7 base JSSP instances of varying size.
- Time-dependent costs: 2 piecewise-constant profiles + 1 randomly drawn profile.
- Cardinality limit $\overline{M}_t \in \{|\mathcal{M}| - 1, |\mathcal{M}| - 2\}$.
- Horizons $|\mathcal{T}| = \lceil \omega C^\star \rceil$:
  - » $\omega \in \{1.2, 1.6, 2.0, 3.0\}$,
  - » $C^\star \in [56, 850]$.

$\Rightarrow$ **504 instances** across small, medium, and large horizon regimes.

Setup: `SCIP` + `PySCIPOpt`, Intel Xeon Gold 5122, 1 thread, 1h TL.

Outline

1. Introduction

2. Branch and Price

3. Computational Experiments
   - 3.1 Setup and Instances
   - 3.2 Main Results
   - 3.3 Ablation Study

4. Conclusion

B&P vs. Compact Formulations

A : Aggregated compact fomulation **D** : Disaggregated compact formulation **E** : Extended formulation

Instances		Branch and Bound (**A**)				Branch and Bound (**D**)				Branch and Price (**E**)			
subset	#inst.	#opt	#feas	T	#n	#opt	#feas	T	#n	#opt	#feas	T	#n
small	144	131	144	94.4	98.5	131	144	134.5	7.0	130	144	15.5	14.3
medium	144	24	99	2226.4	691.7	75	114	1327.3	3.1	110	133	41.2	13.6
large	216	0	54	TL	–	41	78	1813.1	3.0	150	187	79.4	15.3
all	**504**	**155**	**297**	154.4	133.4	**247**	**336**	415.9	4.8	**390**	**464**	**38.5**	14.5

Table: Comparison of B&P against aggregated and disaggregated branch-and-bound.

B&P vs. Compact Formulations

$\mathbf{A}$: Aggregated compact fomulation $\quad$ $\mathbf{D}$: Disaggregated compact formulation $\quad$ $\mathbf{E}$: Extended formulation

Instances		Branch and Bound ($\mathbf{A}$)				Branch and Bound ($\mathbf{D}$)				Branch and Price ($\mathbf{E}$)			
subset	#inst.	#opt	#feas	T	#n	#opt	#feas	T	#n	#opt	#feas	T	#n
small	144	131	144	94.4	98.5	131	144	134.5	7.0	130	144	15.5	14.3
medium	144	24	99	2226.4	691.7	75	114	1327.3	3.1	110	133	41.2	13.6
large	216	0	54	TL	–	41	78	1813.1	3.0	150	187	79.4	15.3
all	**504**	**155**	**297**	154.4	133.4	**247**	**336**	415.9	4.8	**390**	**464**	**38.5**	14.5

Table: Comparison of B&P against aggregated and disaggregated branch-and-bound.

- B&P ($\mathbf{E}$) solves **77%** of all instances to optimality vs. 31% ($\mathbf{A}$) and 49% ($\mathbf{D}$).

B&P vs. Compact Formulations

A : Aggregated compact fomulation **D** : Disaggregated compact formulation **E** : Extended formulation

Instances		Branch and Bound (**A**)				Branch and Bound (**D**)				Branch and Price (**E**)			
subset	#inst.	#opt	#feas	T	#n	#opt	#feas	T	#n	#opt	#feas	T	#n
small	144	131	144	94.4	98.5	131	144	134.5	7.0	130	144	15.5	14.3
medium	144	24	99	2226.4	691.7	75	114	1327.3	3.1	110	133	41.2	13.6
large	216	0	54	TL	–	41	78	1813.1	3.0	150	187	79.4	15.3
all	**504**	**155**	**297**	154.4	133.4	**247**	**336**	415.9	4.8	**390**	**464**	**38.5**	14.5

Table: Comparison of B&P against aggregated and disaggregated branch-and-bound.

- B&P (**E**) solves **77%** of all instances to optimality vs. 31% (**A**) and 49% (**D**).
- B&P explores **small trees** and solves each node **relaxation fast** on all instance subsets.

B&P vs. Compact Formulations

$\mathbf{A}$: Aggregated compact fomulation $\mathbf{D}$: Disaggregated compact formulation $\mathbf{E}$: Extended formulation

Instances		Branch and Bound ($\mathbf{A}$)				Branch and Bound ($\mathbf{D}$)				Branch and Price ($\mathbf{E}$)			
subset	#inst.	#opt	#feas	T	#n	#opt	#feas	T	#n	#opt	#feas	T	#n
small	144	131	144	94.4	98.5	131	144	134.5	7.0	130	144	15.5	14.3
medium	144	24	99	2226.4	691.7	75	114	1327.3	3.1	110	133	41.2	13.6
large	216	0	54	TL	–	41	78	1813.1	3.0	150	187	79.4	15.3
all	**504**	**155**	**297**	154.4	133.4	**247**	**336**	415.9	4.8	**390**	**464**	**38.5**	14.5

Table: Comparison of B&P against aggregated and disaggregated branch-and-bound.

- B&P ($\mathbf{E}$) solves **77%** of all instances to optimality vs. 31% ($\mathbf{A}$) and 49% ($\mathbf{D}$).
- B&P explores **small trees** and solves each node **relaxation fast** on all instance subsets.
- Performance gap **widens with horizon.**

B&P vs. Compact Formulations

$\mathbf{A}$: Aggregated compact fomulation $\mathbf{D}$: Disaggregated compact formulation $\mathbf{E}$: Extended formulation

Instances		Branch and Bound ($\mathbf{A}$)				Branch and Bound ($\mathbf{D}$)				Branch and Price ($\mathbf{E}$)			
subset	#inst.	#opt	#feas	T	#n	#opt	#feas	T	#n	#opt	#feas	T	#n
small	144	131	144	94.4	98.5	131	144	134.5	7.0	130	144	15.5	14.3
medium	144	24	99	2226.4	691.7	75	114	1327.3	3.1	110	133	41.2	13.6
large	216	0	54	TL	–	41	78	1813.1	3.0	150	187	79.4	15.3
all	**504**	**155**	**297**	154.4	133.4	**247**	**336**	415.9	4.8	**390**	**464**	**38.5**	14.5

Table: Comparison of B&P against aggregated and disaggregated branch-and-bound.

- B&P ($\mathbf{E}$) solves **77%** of all instances to optimality vs. 31% ($\mathbf{A}$) and 49% ($\mathbf{D}$).
- B&P explores **small trees** and solves each node **relaxation fast** on all instance subsets.
- Performance gap **widens with horizon.**

B&P vs. Compact Formulations

$\mathbf{A}$: Aggregated compact fomulation $\mathbf{D}$: Disaggregated compact formulation $\mathbf{E}$: Extended formulation

Instances		Branch and Bound ($\mathbf{A}$)				Branch and Bound ($\mathbf{D}$)				Branch and Price ($\mathbf{E}$)			
subset	#inst.	#opt	#feas	T	#n	#opt	#feas	T	#n	#opt	#feas	T	#n
small	144	131	144	94.4	98.5	131	144	134.5	7.0	130	144	15.5	14.3
medium	144	24	99	2226.4	691.7	75	114	1327.3	3.1	110	133	41.2	13.6
large	216	0	54	TL	–	41	78	1813.1	3.0	150	187	79.4	15.3
all	**504**	**155**	**297**	154.4	133.4	**247**	**336**	415.9	4.8	**390**	**464**	**38.5**	14.5

Table: Comparison of B&P against aggregated and disaggregated branch-and-bound.

- B&P ($\mathbf{E}$) solves **77%** of all instances to optimality vs. 31% ($\mathbf{A}$) and 49% ($\mathbf{D}$).
- B&P explores **small trees** and solves each node **relaxation fast** on all instance subsets.
- Performance gap **widens with horizon.**

Outline

Ablation Study: Setup

- Tighter horizons $\omega \in \{1.2, 1.6\}$: 252 instances.
- Configurations:
  - » `basic`: dual stabilization, complementary pricing, and most-dispersed branching rule.
  - » `all`: full configuration.
  - » `all-X`: no primal heuristics.
  - » `all-Ps`: no pseudo-cost and strong branching.
  - » `all-Pr`: no propagation.
- Results on intersection subsets: **all-feas** and **all-opt**.
- Additional metrics:
  - **Primal** (`PI`) and **dual integral** (`DI`) [Berthold, 2013].
  - Time to **first solution** ($\%\mathtt{T}^{\mathtt{1st}}$) and time to **optimal primal bound** ($\%\mathtt{T}^{\mathtt{opt}}$).

Ablation Study: Primal Side

	all inst. (252)		all-feas (166)		all-opt (114)		
variant	#opt	#feas	PI	$\%T^{1st}$	$\%T^{opt}$	T	#n
basic	131	177	24	33	67	117.6	103
all-H	137	172	36	40	84	**72.3**	43
all	**138**	**212**	**17**	**21**	**86**	92.5	**36**

Table: Primal heuristic ablation comparison.

Ablation Study: Primal Side

	all inst. (252)		all-feas (166)		all-opt (114)		
variant	#opt	#feas	PI	$\%T^{1st}$	$\%T^{opt}$	T	#n
basic	131	177	24	33	67	117.6	103
all-H	137	172	36	40	84	**72.3**	43
all	**138**	**212**	**17**	**21**	**86**	92.5	**36**

Table: Primal heuristic ablation comparison.

- **Heuristics** finds feasible solutions **more often and earlier**.

Ablation Study: Primal Side

	all inst. (252)		all-feas (166)		all-opt (114)		
variant	#opt	#feas	PI	$\%T^{1st}$	$\%T^{opt}$	T	#n
`basic`	131	177	24	33	67	117.6	103
`all-H`	137	172	36	40	84	**72.3**	43
`all`	**138**	**212**	**17**	**21**	**86**	92.5	**36**

Table: Primal heuristic ablation comparison.

- **Heuristics** finds feasible solutions **more often and earlier**.
- Per-node overhead: **speed trade-off** on easier instances.

Ablation Study: Primal Side

	all inst. (252)		all-feas (166)		all-opt (114)		
variant	#opt	#feas	PI	$\%T^{1st}$	$\%T^{opt}$	T	#n
basic	131	177	24	33	67	117.6	103
all-H	137	172	36	40	84	**72.3**	43
all	**138**	**212**	**17**	**21**	**86**	92.5	**36**

Table: Primal heuristic ablation comparison.

- **Heuristics** finds feasible solutions **more often and earlier**.
- Per-node overhead: **speed trade-off** on easier instances.
- **Optimality: comparable performance** with and without heuristics.

Ablation Study: Dual Side

	all inst. (252)		all-opt. (115)	
variant	#opt	DI	T	#n
basic	131	3.3	113.8	102
all-Ps	127	**2.3**	147.2	87
all-Pr	**145**	2.6	102.6	52
all	138	2.8	**95.9**	**36**

Table: Dual side features ablation comparison.

Ablation Study: Dual Side

	all inst. (252)		all-opt. (115)	
variant	#opt	DI	T	#n
basic	131	3.3	113.8	102
all-Ps	127	**2.3**	147.2	87
all-Pr	**145**	2.6	102.6	52
all	138	2.8	**95.9**	**36**

Table: Dual side features ablation comparison.

- Pseudo-costs and strong branching: **pruning gains outweigh overhead.**

Ablation Study: Dual Side

	all inst. (252)		all-opt. (115)	
variant	#opt	DI	T	#n
basic	131	3.3	113.8	102
all-Ps	127	**2.3**	147.2	87
all-Pr	**145**	2.6	102.6	52
all	138	2.8	**95.9**	**36**

Table: Dual side features ablation comparison.

- Pseudo-costs and strong branching: **pruning gains outweigh overhead.**
- Propagation: **per-node overhead compounds** across the tree.
  - » Time: **+48%** to pricing and **+70%** to strong branching.

Ablation Study: Dual Side

	all inst. (252)		all-opt. (115)	
variant	#opt	DI	T	#n
basic	131	3.3	113.8	102
all-Ps	127	**2.3**	147.2	87
all-Pr	**145**	2.6	102.6	52
all	138	2.8	**95.9**	**36**

Table: Dual side features ablation comparison.

- Pseudo-costs and strong branching: **pruning gains outweigh overhead.**
- Propagation: **per-node overhead compounds** across the tree.
  - » Time: **+48%** to pricing and **+70%** to strong branching.

→ all: **best** per-node **efficiency**, propagation is the remaining bottleneck.

Outline

1. Introduction
2. Branch and Price
3. Computational Experiments
4. **Conclusion**

Conclusion and outlook

- **Branch-and-price algorithm** for JSSP with time-dependent costs and cardinality constraints.
- **Results on 504 instances**: 77% solved to optimality, outperforming both compact TI formulations.
- Fast pricing and tight original relaxation $\Rightarrow$ **Integrality property is not a limitation**.

Conclusion and outlook

- **Branch-and-price algorithm** for JSSP with time-dependent costs and cardinality constraints.
- **Results on 504 instances**: 77% solved to optimality, outperforming both compact TI formulations.
- Fast pricing and tight original relaxation $\Rightarrow$ **Integrality property is not a limitation**.
- **Outlook**:
  - » Evaluation: benchmark against existing branch-and-cut, time-based objectives.
  - » Algorithm: stronger heuristics, subgradient algorithm, flexible job-shop.

Conclusion and outlook

- **Branch-and-price algorithm** for JSSP with time-dependent costs and cardinality constraints.
- **Results on 504 instances**: 77% solved to optimality, outperforming both compact TI formulations.
- Fast pricing and tight original relaxation ⇒ **Integrality property is not a limitation**.
- **Outlook**:
  - » Evaluation: benchmark against existing branch-and-cut, time-based objectives.
  - » Algorithm: stronger heuristics, subgradient algorithm, flexible job-shop.

marouane-f.github.io

References I

Marouane Felloussi, Xavier Delorme, and Paolo Gianessi. A branch-and-cut algorithm for job-shop scheduling under time-of-use pricing and a peak power limit. *European Journal of Operational Research*, 2026. (to appear).

Marouane Felloussi, Xavier Delorme, and Paolo Gianessi. Minimizing energy cost in a job-shop scheduling problem under ToU pricing: A new method based on a period-indexed MILP. In *14th International Conference on OR and Enterprise Systems*, 2025.

Andreas Bley and Andreas Linß. Propagation and branching strategies for job shop scheduling minimizing the weighted energy consumption. In *International Conference on Operations Research*, pages 573–580. Springer, 2022.

Oussama Masmoudi, Xavier Delorme, and Paolo Gianessi. Job-shop scheduling problem with energy consideration. *International Journal of Production Economics*, 216:12–22, 10 2019. ISSN 09255273. doi: 10.1016/j.ijpe.2019.03.021.

Myoung Ju Park and Andy Ham. Energy-aware flexible job shop scheduling under time-of-use pricing. *International Journal of Production Economics*, 248, 6 2022. ISSN 09255273. doi: 10.1016/j.ijpe.2022.108507.

References II

Enda Jiang and Ling Wang. Multi-objective optimization based on decomposition for flexible job shop scheduling under time-of-use electricity prices. *Knowledge-Based Systems*, 204, 9 2020. ISSN 09507051. doi: 10.1016/j.knosys.2020.106177.

Minh Hung Ho, Faicel Hnaien, and Frederic Dugardin. Exact method to optimize the total electricity cost in two-machine permutation flow shop scheduling problem under time-of-use tariff. *Computers and Operations Research*, 144, 8 2022. ISSN 03050548. doi: 10.1016/j.cor.2022.105788.

Mauro Gaggero, Massimo Paolucci, and Roberto Ronco. Exact and heuristic solution approaches for energy-efficient identical parallel machine scheduling with time-of-use costs. *European Journal of Operational Research*, 311:845–866, 12 2023. ISSN 03772217. doi: 10.1016/j.ejor.2023.05.040.

Ada Che, Shibohua Zhang, and Xueqi Wu. Energy-conscious unrelated parallel machine scheduling under time-of-use electricity tariffs. *Journal of Cleaner Production*, 156:688–697, 7 2017. ISSN 09596526. doi: 10.1016/j.jclepro.2017.04.018.

Jian Ya Ding, Shiji Song, Rui Zhang, Raymond Chiong, and Cheng Wu. Parallel machine scheduling under time-of-use electricity prices: New models and optimization approaches. *IEEE Transactions on Automation Science and Engineering*, 13:1138–1154, 4 2016. ISSN 15455955. doi: 10.1109/TASE.2015.2495328.

References III

Zheng Tian and Li Zheng. Single machine parallel-batch scheduling under time-of-use electricity prices: New formulations and optimisation approaches. *European Journal of Operational Research*, 312: 512–524, 1 2024. ISSN 03772217. doi: 10.1016/j.ejor.2023.07.012.

Junheng Cheng, Feng Chu, Chengbin Chu, and Weili Xia. Bi-objective optimization of single-machine batch scheduling under time-of-use electricity prices. *RAIRO - Operations Research*, 50:715–732, 10 2016. ISSN 28047303. doi: 10.1051/ro/2015063.

U.S. EIA. Sources of price volatility in the ERCOT market. Technical report, U.S. Energy Information Administration, 2022. URL `https://www.eia.gov/outlooks/steo/special/supplements/2022/2022_sp_03.pdf`.

Nord Pool. Transition to 15-minute market time unit (MTU), 2025. URL `https://www.nordpoolgroup.com/en/trading/transition-to-15-minute-market-time-unit-mtu/`.

Christian Artigues. On the strength of time-indexed formulations for the resource-constrained project scheduling problem. *Operations Research Letters*, 2017. ISSN 0167-6377.

Andreas Drexl and Alf Kimms. Optimization guided lower and upper bounds for the resource investment problem. *Journal of the Operational Research Society*, 2001.

References IV

Giuseppe Lancia, Franca Rinaldi, and Paolo Serafini. A time-indexed LP-based approach for min-sum job-shop problems. *Annals of Operations Research*, 186(1):175–198, 2011.

Maximilian Kolter, Martin Grunow, and Rainer Kolisch. On branch-and-price for multi-project scheduling. In *19th International Workshop on Project Management and Scheduling*, pages 69–72, Apr 2024.

A Pessoa, R Sadykov, E Uchoa, and F Vanderbeck. Automation and combination of linear-programming based stabilization techniques in column generation. *INFORMS J. Comput.*, 30(2):339–360, May 2018.

Anuj Mehrotra and Michael A Trick. A column generation approach for graph coloring. *INFORMS J. Comput.*, 8(4):344–354, 1996.

Marco E. Lübbecke. Column generation. In *Wiley Encyclopedia of Operations Research and Management Science*. John Wiley & Sons, Ltd, 2011.

Timo Berthold. Measuring the impact of primal heuristics. *Operations Research Letters*, 41(6):611–614, 2013. ISSN 0167-6377.

Appendix: Dynamic Programming Recurrence

Let $d_j^\star(i,t)$ = minimum cost of any partial schedule completing operation i at time t:

$$d_j^\star(i,t) = \min\Big\{\underbrace{d_j^\star(i,t-1)}_{\text{idle}},\quad \underbrace{d_j^\star(i-1,\,t-q_i) + S_i^t}_{\text{processing}}\Big\}.$$

- Minimum reduced cost of any complete j-schedule: $d_j^\star(|\mathcal{M}|,|\mathcal{T}|)$.

Appendix: Pricing improvements

- **Dual stabilization**: parameter-free scheme of Pessoa et al. [2018] based on sub-gradient information $\Rightarrow$ reduces dual oscillations and speeds convergence.
- For any dual optimal solution of **(MP)**: $\alpha_j \geqslant \min_{p \in \mathcal{P}_j} c_p \Rightarrow$ enforced in the dual space as a variable in the primal.
- **Complementary pricing**: at each root-node CG iteration, an additional randomized pricing call forbids (j, m, t) with probability $\sum_{p \in \mathcal{P}_j} a^p_{m,t} \hat{\lambda}_p$, generating columns complementary to the active ones.
- **Window sums** S_i^t updated in $O(1)$ per step: $S_i^t = S_i^{t-1} - \hat{c}_i^{t-q_i} + \hat{c}_i^t$.
- **Time windows** from precedences restrict admissible (i, t) states $\Rightarrow$ arc fixings.

Appendix: Lagrangian Bound and Exact Branching

- With **exact pricing**, compute a **Lagrangian lower bound** LB_{lag} on the value of **(MP)** Lübbecke [2011].
- Data scaled to obtain an integer objective $\Rightarrow \lceil LB_{\text{lag}} \rceil$ is a valid lower bound.
- **Early branching** Mehrotra and Trick [1996]: branch before CG convergence if

$$\lceil z_{\text{RMP}} \rceil = \lceil LB_{\text{lag}} \rceil,$$

  without losing relaxation strength.
- Saves LP re-optimization iterations at nodes where the bound is already tight.

Appendix: Branching in the ISP

- Branch on original binary variables:

$$z_{j,m}^{t} = \sum_{p \in \mathcal{P}_j} a_{m,t}^{p} \lambda_p \in \{0, 1\},$$

  indicating whether (j, m) is processing at time t.

- Integrality of $\mathbf{z} \iff$ integrality of $\boldsymbol{\lambda}$ (set-partitioning argument).

- **Two branches**:
  - » $z_{j,m}^{t} = 0$: forbid processing transition of (j, m) at t in DP.
  - » $z_{j,m}^{t} = 1$: force (j, m) at t, forbid all $t' \neq t \Rightarrow$ also reduces to arc fixings in DP.

- Branching on master variables (resource, disjunction) led to large trees in preliminary experiments.

Appendix: Hierarchical Branching Scheme

- **Step 1 — Most-dispersed rule**: select the operation (j, m) whose execution is most spread in the fractional solution:

  $$(j, m) = \arg\max_{(j,m)} |F_{j,m}|, \quad F_{j,m} := \{t : 0 < z^{\star t}_{j,m} < 1\},$$

  then branch on $t = \arg\max_{t \in F_{j,m}} z^{\star t}_{j,m}$.
- **Step 2 — Strong branching** (near root): on the candidates from Step 1, solve the **(RMP)** for each candidate branch without re-pricing; pick the best dual bound improvement.
- **Step 3 — Pseudo-cost branching**: once enough observations are collected, use pseudo-costs (complemented with aggregated operation- and time-level scores) to select the branching variable.
- Falls back to most-dispersed rule if no reliable pseudo-costs are available.

Appendix: Propagation

Fixings derived from branching decisions $\mathcal{B}_n$:

- **Job/machine disjunction**: if $z^t_{j,m} = 1$, then $z^t_{j,m'} = 0$ for all $m' \neq m$, and $z^t_{j',m} = 0$ for all $j' \neq j$.
- **Precedence**: if $z^t_{j,m} = 1$, fix $z^{t'}_{j,m'} = 0$ for $(j, m') \prec (j, m)$, $t' > t$ (and symmetrically for successors).
- **Non-preemption (contiguity)**: if $z^{t_1}_{j,m} = z^{t_2}_{j,m} = 1$, then $z^t_{j,m} = 1$ for all $t_1 < t < t_2$.
- **Non-preemption (interruption)**: if $z^{t_1}_{j,m} = 0$, $z^{t_2}_{j,m} = 1$, fix $z^t_{j,m} = 0$ for $t < t_1$ (when $t_1 < t_2$).
- **Cardinality limit**: if $\overline{M}_t$ machines are already fixed active at t, fix remaining machines idle.

Propagation is also invoked during strong branching.

Appendix: Exact Tree Search

- 0-1 RMP is large and highly constrained.
- **Depth-first search**: column assignment per job in a predefined order of traversal.
- A node is **pruned** by comparing incumbent value against
  - Cost of partial solution at a node, or
  - LP-free additive lower bound $\mathtt{LB}$
    - » $\mathtt{LB} :=$ partial cost $+ \sum$ costs of cheapest compatible columns among remaining jobs

Appendix: Large Neighborhood Search

- Infeasible $\Rightarrow$ repair (find solution)
  - **Repair**: reoptimize unfixed jobs in partial solutions.
  - Same oracle as pricing, with different objective and fixings.
- Feasible $\Rightarrow$ destroy-repair (improve solution)
  - A **destroy** set allows to define a neighborhood around an incumbent solution.
  - Run pricing oracle over the neighborhood.

Appendix: Root LP Statistics

- Over all 504 instances, within the time limit:
  - » $\mathbf{E}$: **499** root relaxations solved.
  - » $\mathbf{A}$: **391** root relaxations solved.
  - » $\mathbf{D}$: **338** root relaxations solved.
- Average LP gap (root bound vs. optimal), computed over available instances:
  - » $\mathbf{E}$: **0.01%** $\mathbf{D}$: **0.01%** $\mathbf{A}$: **0.03%**
- $\mathbf{E}$ and $\mathbf{D}$ share the same LP relaxation strength (integrality property).
- Key insight: $\mathbf{D}$ finds more feasible/optimal solutions than $\mathbf{A}$, but fails at scale:
  - » 54 of the 168 infeasible $\mathbf{D}$ runs terminate due to **memory limits**.
  - » 66 fail to solve the **root LP** within the time limit on medium instances.