ROADEF 2026
A GNN-aided fix-and-reoptimize approach for the Lot Sizing Problem

Mathieu Lerouge[1,2]
Andrea Lodi[2], Enrico Malaguti[2], Michele Monaci[2] & Filippo Focacci[1]
mathieu.lerouge@decisionbrain.com mathieulerouge.github.io

February 24, 2026

1 DecisionBrain™ Smarter Decisions. Better Results.

2 ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

Plan

Plan

Lot Sizing Problem (LSP)

Given:

- a planning horizon discretized into T **periods**;
- a set of M **machines** with limited capacities;
- a set of N **items**, such that each item has:
  - demands over time periods,
  - ...
  - setup, production, inventory and lost sales unitary costs;

define a **production plan** minimizing the total cost (setup, production, inventory and lost sales).

Lot Sizing Problem (LSP)

Given:

- a planning horizon discretized into T **periods**;
- a set of M **machines** with limited capacities;
- a set of N **items**, such that each item has:
  - demands over time periods,
  - ...
  - setup, production, inventory and lost sales unitary costs;

define a **production plan** minimizing the total cost (setup, production, inventory and lost sales).

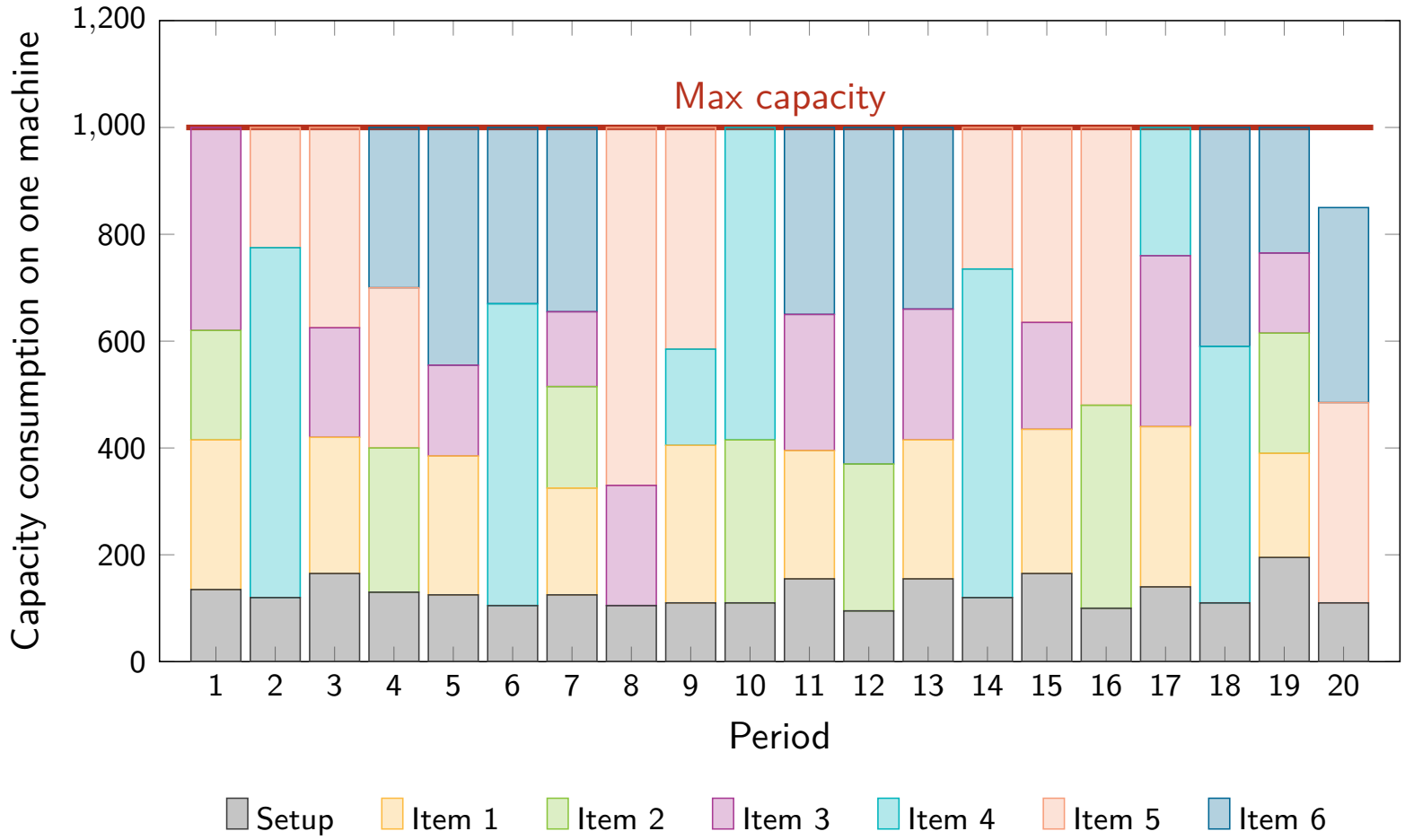
Lot Sizing Problem (LSP)

Given:

- a planning horizon discretized into T **periods**;
- a set of M **machines** with limited capacities;
- a set of N **items**, such that each item has:
  - demands over time periods,
  - ...
  - setup, production, inventory and lost sales unitary costs;

define a **production plan** minimizing the total cost (setup, production, inventory and lost sales).

LSP solution



LSP MILP model

Typically, near-optimal solutions are obtained by solving a **MILP** for a **long time** (*e.g.* hours):

min setup + production + inventory + lost sales costs

s.t. balance constraints (prod. + inventory vs demand + lost sales)

capacity constraints

minimum production constraints

[...]

$Y_{ijt} \in \{0,1\} \quad i \in \{\text{items}\},\ j \in \{\text{machines}\},\ t \in \{\text{periods}\}$

[...]

LSP MILP model

Typically, near-optimal solutions are obtained by solving a **MILP** for a **long time** (*e.g.* hours):

min setup + production + inventory + lost sales costs

s.t. balance constraints (prod. + inventory vs demand + lost sales)

capacity constraints

minimum production constraints

[...]

$Y_{ijt} \in \{0,1\} \quad i \in \{\text{items}\},\ j \in \{\text{machines}\},\ t \in \{\text{periods}\}$

[...]

LSP MILP model

Typically, near-optimal solutions are obtained by solving a **MILP** for a **long time** (*e.g.* hours):

min setup + production + inventory + lost sales costs

s.t. balance constraints (prod. + inventory vs demand + lost sales)

capacity constraints

minimum production constraints

[...]

$Y_{ijt} \in \{0,1\} \quad i \in \{\text{items}\},\ j \in \{\text{machines}\},\ t \in \{\text{periods}\}$

[...]

LSP MILP model

Typically, near-optimal solutions are obtained by solving a **MILP** for a **long time** (*e.g.* hours):

min setup + production + inventory + lost sales costs

s.t. balance constraints (prod. + inventory vs demand + lost sales)

capacity constraints

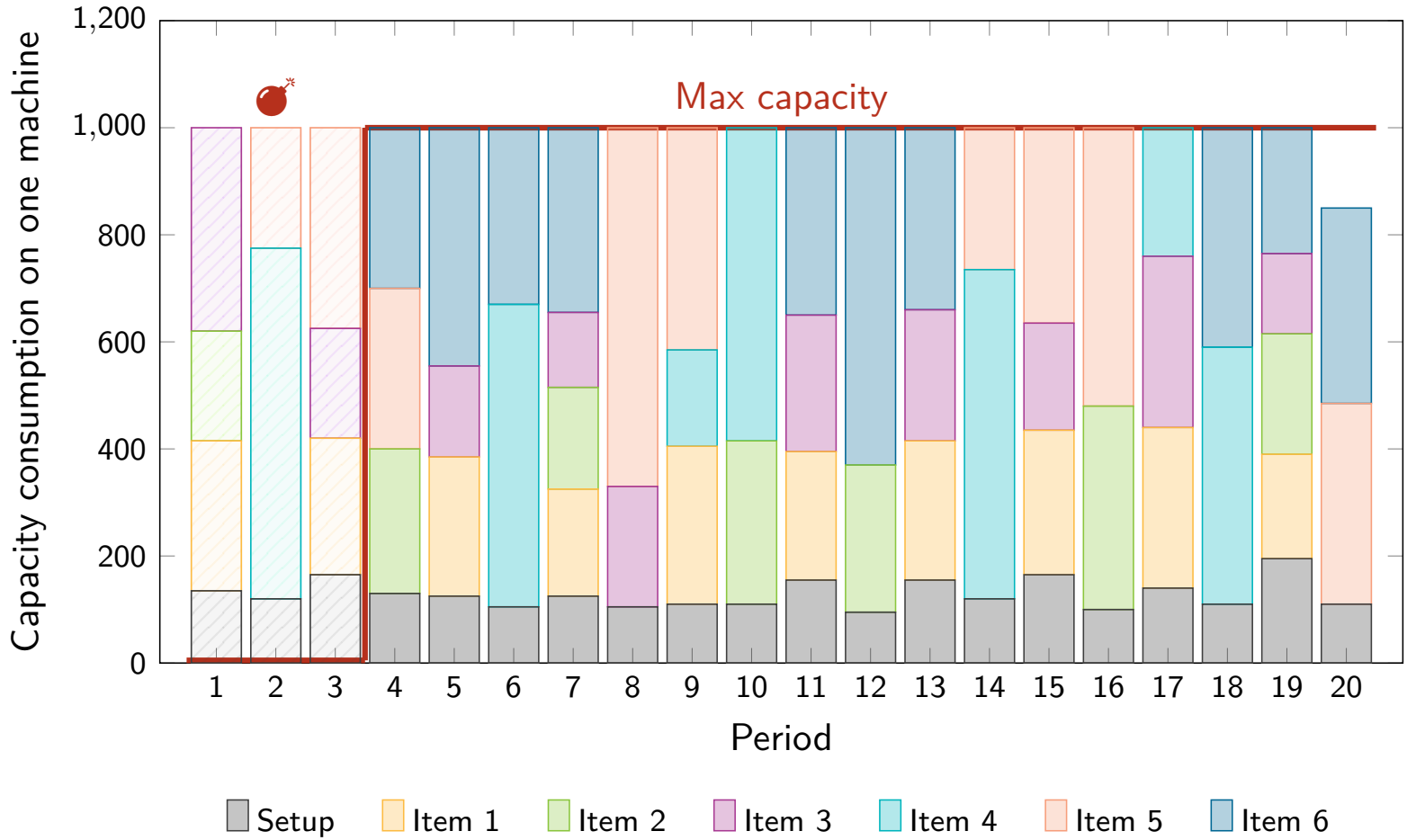
minimum production constraints

[...]

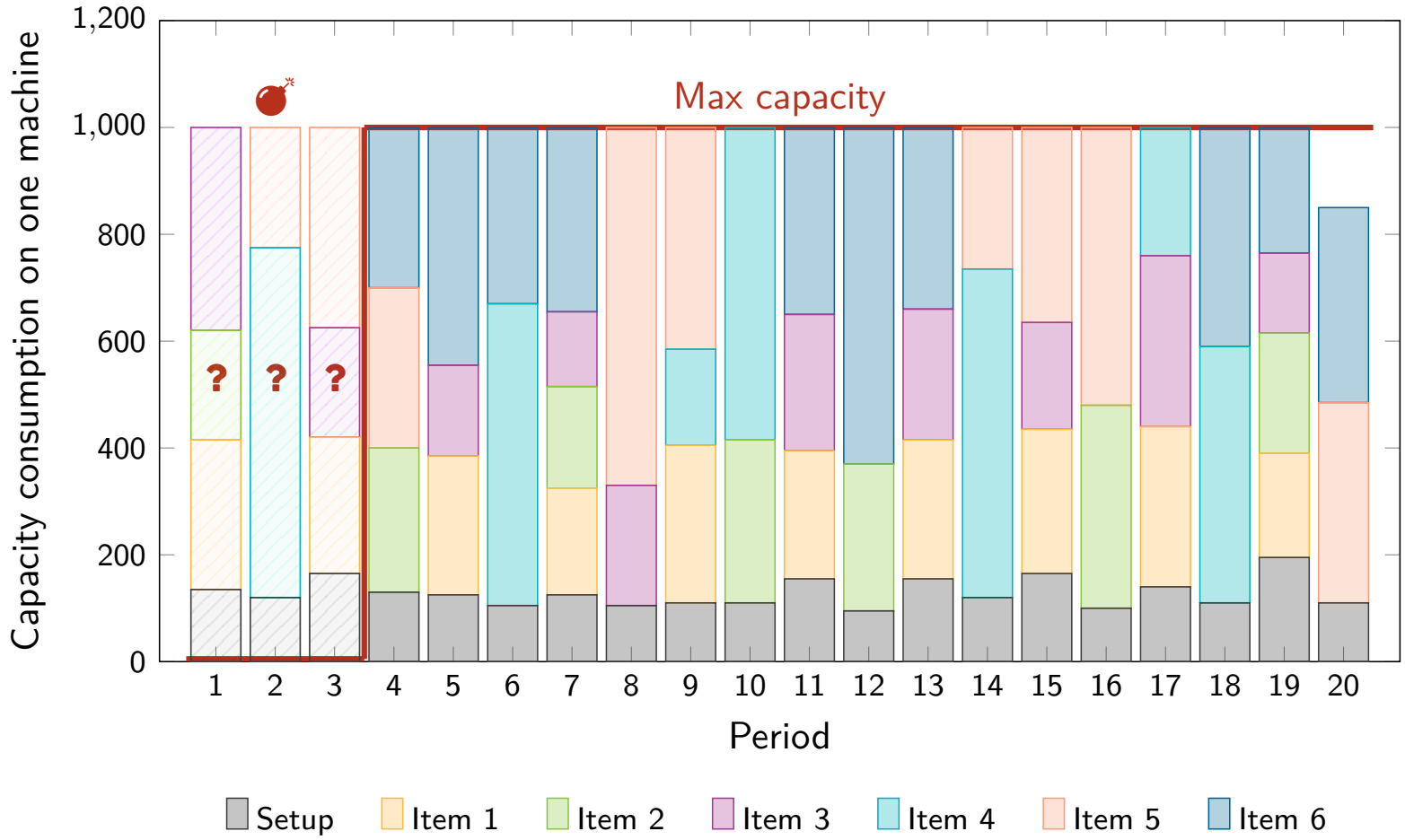
$Y_{ijt} \in \{0,1\} \quad i \in \{\text{items}\},\ j \in \{\text{machines}\},\ t \in \{\text{periods}\}$

[...]

LSP disruption



LSP disruption



How can we react?

Solve the **MILP model on perturbed instance:**

- ✔ Optimal / high-quality solution;
- ✖ Potentially far from original solution;
- ✖ Computationally prohibitive.

Apply **simple repair heuristic** to obtain a repaired solution $\mathcal{S}^r$ (by canceling productions on the disrupted machine(s)):

- ✔ Extremely fast to compute;
- ✔ Close to the original solution;
- ✖ Poor-quality solution (massive lost sales costs).

Maybe $\mathcal{S}^r$ can be used as a starting point of a reoptimization process?

How can we react?

Solve the **MILP model on perturbed instance:**

- ✔ Optimal / high-quality solution;
- ✖ Potentially far from original solution;
- ✖ Computationally prohibitive.

Apply **simple repair heuristic** to obtain a repaired solution $\mathcal{S}^r$ (by canceling productions on the disrupted machine(s)):

- ✔ Extremely fast to compute;
- ✔ Close to the original solution;
- ✖ Poor-quality solution (massive lost sales costs).

Maybe $\mathcal{S}^r$ can be used as a starting point of a reoptimization process?

How can we react?

Solve the **MILP model on perturbed instance:**

- ✔ Optimal / high-quality solution;
- ✖ Potentially far from original solution;
- ✖ Computationally prohibitive.

Apply **simple repair heuristic** to obtain a repaired solution $\mathcal{S}^r$ (by canceling productions on the disrupted machine(s)):

- ✔ Extremely fast to compute;
- ✔ Close to the original solution;
- ✖ Poor-quality solution (massive lost sales costs).

Maybe $\mathcal{S}^r$ can be used as a starting point of a reoptimization process?

Need

Given an instance, a (nominal) solution $\mathcal{S}^o$ and a disruption, compute a **new solution** $\mathcal{S}^n$ that satisfies four key criteria:

(a) **feasibility** of $\mathcal{S}^n$ w.r.t. the disruption;

(b) **good quality** of $\mathcal{S}^n$;

(c) **fast computation** (*e.g.* seconds $\neq$ hours);

(d) **controlled deviation** of $\mathcal{S}^n$ from $\mathcal{S}^o$.

How to compute such a new solution $\mathcal{S}^n$?

- using an MILP-based approach;
- involving Machine Learning.

Need

Given an instance, a (nominal) solution $\mathcal{S}^o$ and a disruption, compute a **new solution** $\mathcal{S}^n$ that satisfies four key criteria:

(a) **feasibility** of $\mathcal{S}^n$ w.r.t. the disruption;

(b) **good quality** of $\mathcal{S}^n$;

(c) **fast computation** (*e.g.* seconds $\neq$ hours);

(d) **controlled deviation** of $\mathcal{S}^n$ from $\mathcal{S}^o$.

How to compute such a new solution $\mathcal{S}^n$?

- using an MILP-based approach;
- involving Machine Learning.

Related works - ML for MILP reoptimization

Existing approaches:

Using ML to reduce feasible solution space (*e.g.* predicting bounds).

[Lodi et al., 2020] Learning to handle parameter perturbations [...]

[Morabit et al., 2024] Learning to repeatedly solve routing problems

Limitation:

Relying on standard ML models (Random Forests, MLPs) that require **fixed-size input vectors**.

↪ Cannot handle instances or disruptions of varying dimensions.

Our choice: Graph Neural Networks (GNNs)

- Adapted to handle inputs of varying dimensions;
- Able to capture rich relational structures of MILPs.

[Gasse et al., 2019] Exact combinatorial optimization with GCNNs

Related works - ML for MILP reoptimization

Existing approaches:

Using ML to reduce feasible solution space (*e.g.* predicting bounds).

[Lodi et al., 2020] Learning to handle parameter perturbations [...]

[Morabit et al., 2024] Learning to repeatedly solve routing problems

Limitation:

Relying on standard ML models (Random Forests, MLPs) that require **fixed-size input vectors**.

↪ Cannot handle instances or disruptions of varying dimensions.

Our choice: Graph Neural Networks (GNNs)

- Adapted to handle inputs of varying dimensions;
- Able to capture rich relational structures of MILPs.

[Gasse et al., 2019] Exact combinatorial optimization with GCNNs

Related works - ML for MILP reoptimization

Existing approaches:

Using ML to reduce feasible solution space (*e.g.* predicting bounds).

[Lodi et al., 2020] Learning to handle parameter perturbations [...]

[Morabit et al., 2024] Learning to repeatedly solve routing problems

Limitation:

Relying on standard ML models (Random Forests, MLPs) that require **fixed-size input vectors**.

↪ Cannot handle instances or disruptions of varying dimensions.

Our choice: Graph Neural Networks (GNNs)

- Adapted to handle inputs of varying dimensions;
- Able to capture rich relational structures of MILPs.

[Gasse et al., 2019] Exact combinatorial optimization with GCNNs

Plan

Neighborhood constraint

To control the deviation from $\mathcal{S}^r$, we introduce:

- $\tau \leq T$ defining the **short-term horizon** (first τ periods);
- $\kappa \ll N \cdot M \cdot \tau$ **maximal number** of setup variable **changes**.

At most κ setup variables within the short-term horizon are allowed to flip their value from $\mathcal{S}^r$:

$$\sum_{i=1}^{N}\sum_{j=1}^{M}\Bigg(\sum_{\substack{t=1 \\ \text{s.t.} \\ Y^r_{ijt}=1}}^{\tau}(1-Y_{ijt}) + \sum_{\substack{t=1 \\ \text{s.t.} \\ Y^r_{ijt}=0}}^{\tau} Y_{ijt}\Bigg) \leq \kappa$$

Neighborhood constraint

To control the deviation from $\mathcal{S}^r$, we introduce:

- $\tau \leq T$ defining the **short-term horizon** (first τ periods);
- $\kappa \ll N \cdot M \cdot \tau$ **maximal number** of setup variable **changes**.

At most κ setup variables within the short-term horizon are allowed to flip their value from $\mathcal{S}^r$:

$$\sum_{i=1}^{N}\sum_{j=1}^{M}\Bigg(\sum_{\substack{t=1 \\ \text{s.t.} \\ Y^r_{ijt}=1}}^{\tau}(1-Y_{ijt}) + \sum_{\substack{t=1 \\ \text{s.t.} \\ Y^r_{ijt}=0}}^{\tau} Y_{ijt}\Bigg) \leq \kappa$$

GNN-aided fix-and-optimize strategy

Reoptimization MILP model:

LSP MILP + Neighborhood constraint.

GNN-aided strategy:

Solving the reoptimization MILP model (using $\mathcal{S}^r$ as a warm-start) with a **solution space reduced** by:

- identifying $\mathcal{Y}$ a small subset of promising short-term setup variables to be **kept free** in the reoptimization MILP model;
- and **hard-fixing** remaining short-term setup variables

$$Y_{ijt} = Y^r_{ijt} \quad \forall\, i \in [\![1, N]\!],\ \forall\, j \in [\![1, M]\!],\ \forall\, t \in [\![1, \tau]\!],\ (i, j, t) \notin \mathcal{Y}$$

Baseline strategy

Solving the reoptimization MILP model (using $\mathcal{S}^r$ as a warm-start).

GNN-aided fix-and-optimize strategy

Reoptimization MILP model:

LSP MILP + Neighborhood constraint.

GNN-aided strategy:

Solving the reoptimization MILP model (using $\mathcal{S}^r$ as a warm-start) with a **solution space reduced** by:

- identifying $\mathcal{Y}$ a small subset of promising short-term setup variables to be **kept free** in the reoptimization MILP model;
- and **hard-fixing** remaining short-term setup variables

$$Y_{ijt} = Y^r_{ijt} \quad \forall\, i \in [\![1, N]\!],\ \forall\, j \in [\![1, M]\!],\ \forall\, t \in [\![1, \mathcal{T}]\!],\ (i, j, t) \notin \mathcal{Y}$$

Baseline strategy

Solving the reoptimization MILP model (using $\mathcal{S}^r$ as a warm-start).

GNN-aided fix-and-optimize strategy

Reoptimization MILP model:

LSP MILP + Neighborhood constraint.

GNN-aided strategy:

Solving the reoptimization MILP model (using $\mathcal{S}^r$ as a warm-start) with a **solution space reduced** by:

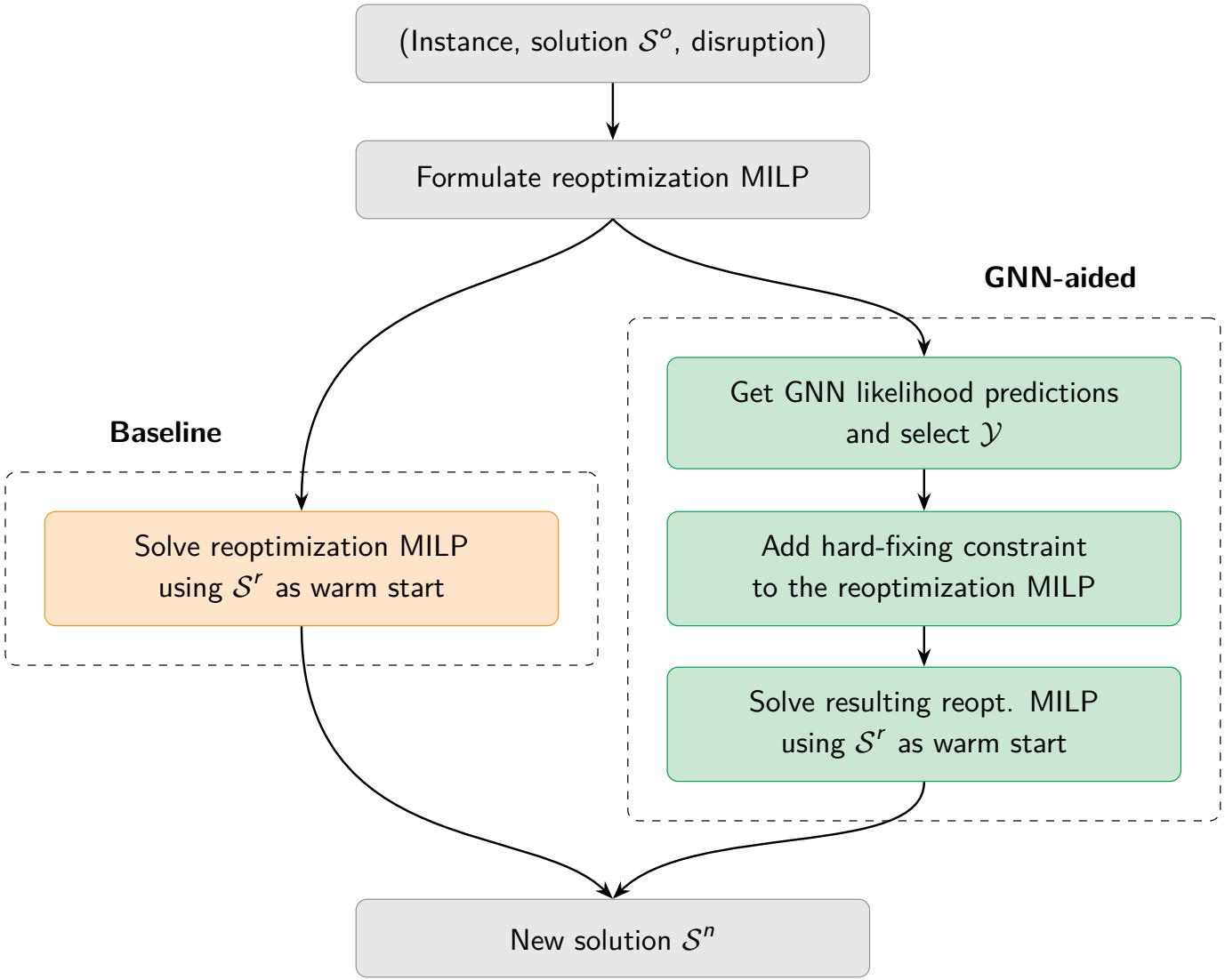
- identifying $\mathcal{Y}$ a small subset of promising short-term setup variables to be **kept free** in the reoptimization MILP model;
- and **hard-fixing** remaining short-term setup variables

$$Y_{ijt} = Y^r_{ijt} \quad \forall\, i \in [\![1, N]\!],\ \forall\, j \in [\![1, M]\!],\ \forall\, t \in [\![1, \mathcal{T}]\!],\ (i, j, t) \notin \mathcal{Y}$$

Baseline strategy

Solving the reoptimization MILP model (using $\mathcal{S}^r$ as a warm-start).

Baseline & GNN-aided strategies



Plan

Encoding data into a feature graph

Given a triplet (instance, nominal solution $\mathcal{S}^o$, disruption), we build a **feature graph**.

Nodes:

- **Machine-Period (MP):** $M \cdot T$ nodes $\{(j,t)\}$;
- **Item-Period (IP):** $N \cdot T$ nodes $\{(i,t)\}$;
- **Production (Pr):** $N \cdot M \cdot T$ nodes $\{(i,j,t)\}$;

where each node is associated with features.

Edges:

- **Time:** link the same entity across consecutive periods;
- **Resource:** connect a Pr node to its MP and IP nodes;
- **Competition:** connect competing Pr nodes.

Encoding data into a feature graph

Given a triplet (instance, nominal solution $\mathcal{S}^o$, disruption), we build a **feature graph**.

Nodes:

- **Machine-Period (MP):** $M \cdot T$ nodes $\{(j,t)\}$;
- **Item-Period (IP):** $N \cdot T$ nodes $\{(i,t)\}$;
- **Production (Pr):** $N \cdot M \cdot T$ nodes $\{(i,j,t)\}$;

where each node is associated with features.

Edges:

- **Time:** link the same entity across consecutive periods;
- **Resource:** connect a Pr node to its MP and IP nodes;
- **Competition:** connect competing Pr nodes.

Encoding data into a feature graph

Given a triplet (instance, nominal solution $\mathcal{S}^o$, disruption), we build a **feature graph**.

Nodes:

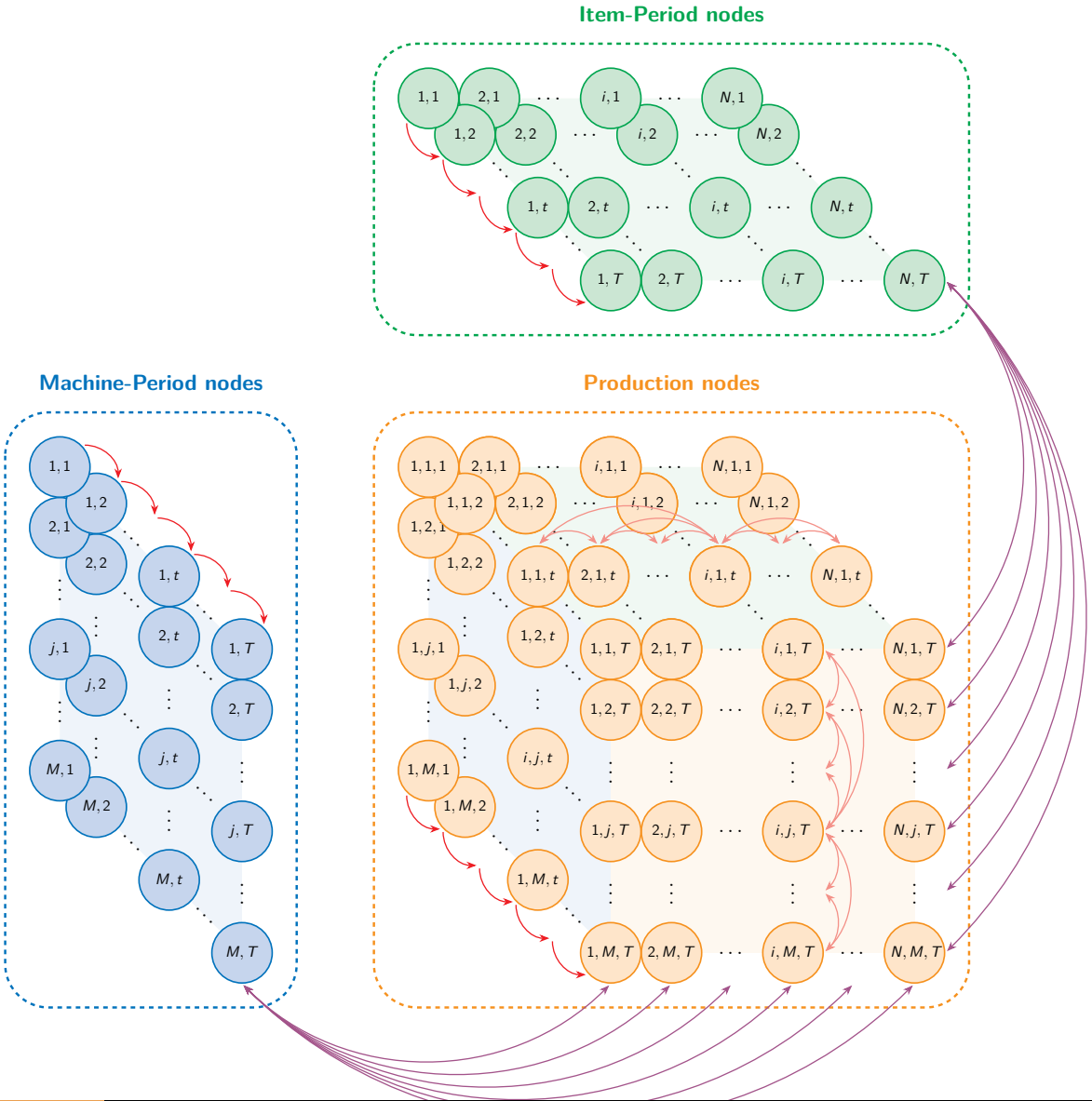
- **Machine-Period (MP):** $M \cdot T$ nodes $\{(j,t)\}$;
- **Item-Period (IP):** $N \cdot T$ nodes $\{(i,t)\}$;
- **Production (Pr):** $N \cdot M \cdot T$ nodes $\{(i,j,t)\}$;

where each node is associated with features.

Edges:

- **Time:** link the same entity across consecutive periods;
- **Resource:** connect a Pr node to its MP and IP nodes;
- **Competition:** connect competing Pr nodes.

Feature graph



GNN processing

Input:

Feature graph encoding (instance, nominal solution $\mathcal{S}^o$, disruption).

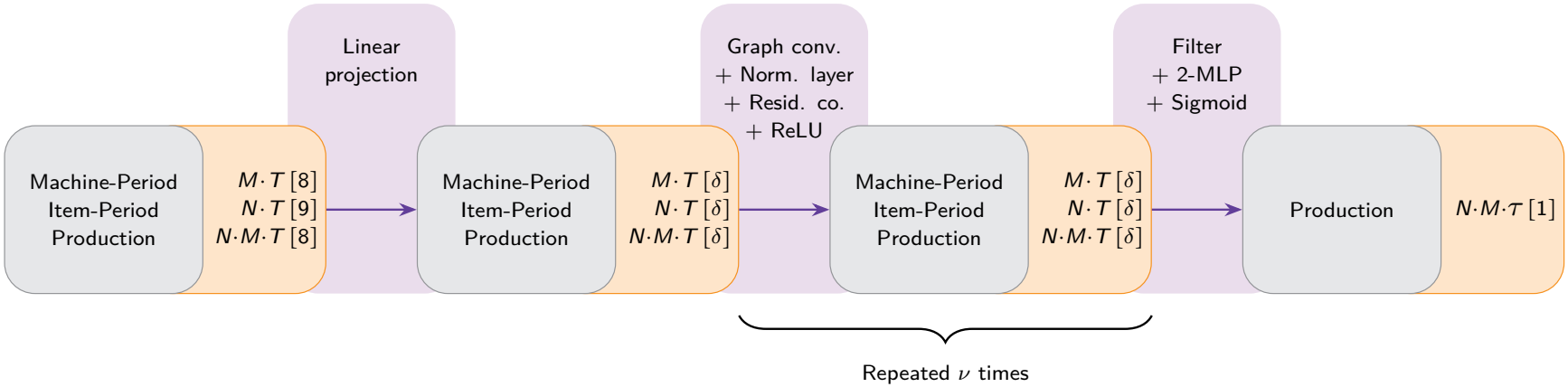
Output:

For each short-term setup variable Y_{ijt}, a score $\sigma_{ijt} \in [0, 1]$ representing the likelihood that Y_{ijt} should change its value.
We select the variable with the best scores to form $\mathcal{Y}$.

GNN processing

Input:

Feature graph encoding (instance, nominal solution $\mathcal{S}^o$, disruption).



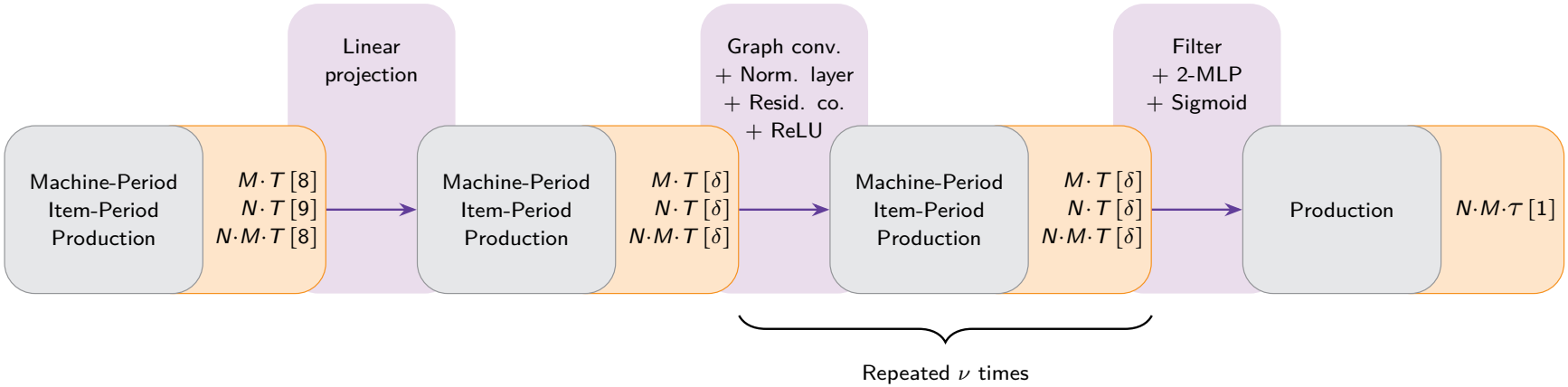
Output:

For each short-term setup variable Y_{ijt}, a score $\sigma_{ijt} \in [0, 1]$ representing the likelihood that Y_{ijt} should change its value.
We select the variable with the best scores to form $\mathcal{Y}$.

GNN processing

Input:

Feature graph encoding (instance, nominal solution $\mathcal{S}^o$, disruption).



Output:

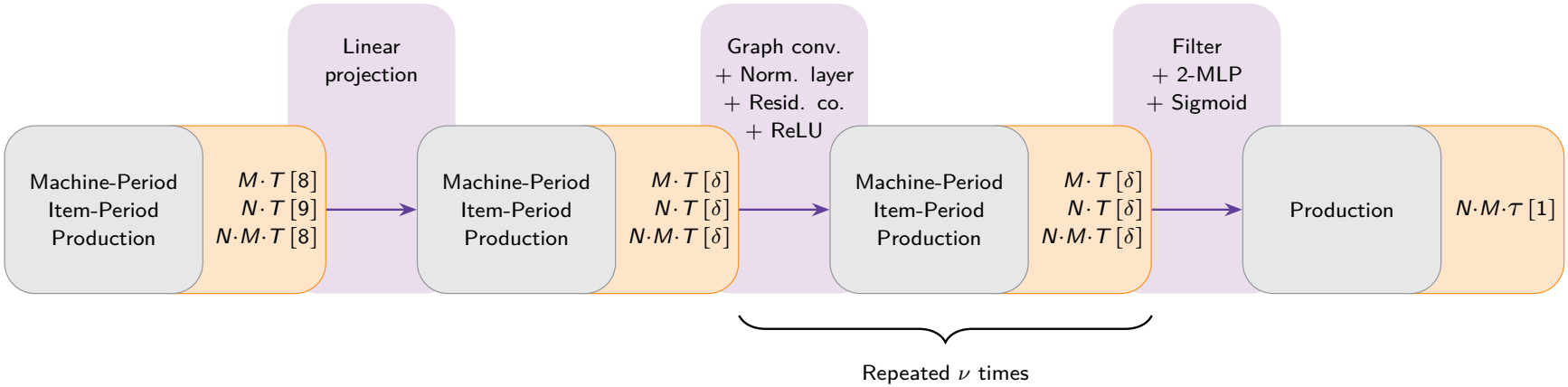
For each short-term setup variable Y_{ijt}, a score $\sigma_{ijt} \in [0, 1]$ representing the likelihood that Y_{ijt} should change its value.

We select the variable with the best scores to form $\mathcal{Y}$.

GNN processing

Input:

Feature graph encoding (instance, nominal solution $\mathcal{S}^o$, disruption).



Output:

For each short-term setup variable Y_{ijt}, a score $\sigma_{ijt} \in [0, 1]$ representing the likelihood that Y_{ijt} should change its value.

We select the variable with the best scores to form $\mathcal{Y}$.

GNN training

To train the GNN, we need **labeled data**.

Given a triplet, we compute its label as follows:

- Solve the reoptimization MILP with a **long time limit** to get a **near-optimal solution** $\mathcal{S}^\star$;
- Associate label $y = 1$ to every short-term **setup variable** that **changes** values between $\mathcal{S}^r$ and $\mathcal{S}^\star$.

Plan

Implementation environment

Software & Hardware:

- Python, Gurobi 12.0.2, PyTorch / PyTorch Geometric;
- AMD Ryzen Threadripper 7955WX PRO 16-Core CPU, 64 GB RAM, NVIDIA RTX A400 GPU.

Computational metric:

To ensure deterministic solver behavior and reproducibility (by isolating the solver's performance from system-level variability), use of Gurobi **work units** (wu).

- **Long time** limit: 6000wu (for nominal solution $\mathcal{S}^o$ and near-optimal solution $\mathcal{S}^\star$);
- **Short time** limit: 10wu (for new solution $\mathcal{S}^n$).

Implementation environment

Software & Hardware:

- Python, Gurobi 12.0.2, PyTorch / PyTorch Geometric;
- AMD Ryzen Threadripper 7955WX PRO 16-Core CPU, 64 GB RAM, NVIDIA RTX A400 GPU.

Computational metric:

To ensure deterministic solver behavior and reproducibility (by isolating the solver's performance from system-level variability), use of Gurobi **work units** (wu).

- **Long time** limit: 6000wu (for nominal solution $\mathcal{S}^o$ and near-optimal solution $\mathcal{S}^\star$);
- **Short time** limit: 10wu (for new solution $\mathcal{S}^n$).

Instance dataset

Two sets of instances (uniform vs. varying dimensions):

Instance		Number of			Machine-item
Set	**Count**	**Machines**	**Items**	**Periods**	**Incompatibilities**
Set 1	250	3	30	30	None
Set 2	250	$\{2, 3, 4\}$	$\{30, 35, 40\}$	30	1 per item

In each set, **three categories of items**:

Priority	Item count	Demand w.r.t. capacity	Unit inv. cost	Unit lost sales cost
High	6 – 8	40% – 50%	$[0.05, 0.15]$	$[5, 9] \rightarrow [0.1, 0.2]$
Medium	9 – 11	40% – 50%	$[0.20, 0.30]$	$[0.9, 1.1] \rightarrow [0.1, 0.2]$
Low	remainder	20% – 30%	$[0.05, 0.35]$	$[0.9, 1.1] \rightarrow [0.1, 0.2]$

Instance dataset

Two sets of instances (uniform vs. varying dimensions):

Instance		Number of			Machine-item
Set	**Count**	**Machines**	**Items**	**Periods**	**Incompatibilities**
Set 1	250	3	30	30	None
Set 2	250	$\{2, 3, 4\}$	$\{30, 35, 40\}$	30	1 per item

In each set, **three categories of items**:

Priority	**Item count**	**Demand w.r.t. capacity**	**Unit inv. cost**	**Unit lost sales cost**
High	6 – 8	40% – 50%	$[0.05, 0.15]$	$[5, 9] \rightarrow [0.1, 0.2]$
Medium	9 – 11	40% – 50%	$[0.20, 0.30]$	$[0.9, 1.1] \rightarrow [0.1, 0.2]$
Low	remainder	20% – 30%	$[0.05, 0.35]$	$[0.9, 1.1] \rightarrow [0.1, 0.2]$

Nominal solutions and disruptions

Nominal solutions:

Solving the LSP MILP for a long time budget (6000 wu).

Disruptions:

- **Machine Breakdown (MB):** 1 mach. unavail. for 4-5 periods;
- **Plant Shutdown (PS):** All machines unavail. for 1-2 periods.

Disruption impact on nominal solution:

By repairing $\mathcal{S}^o$ into $\mathcal{S}^r$:

- Only ~4% of setup variables are modified;
- But total cost doubles on average (+101% cost increase), due to massive lost sales penalties.

Nominal solutions and disruptions

Nominal solutions:

Solving the LSP MILP for a long time budget (6000 wu).

Disruptions:

- **Machine Breakdown (MB):** 1 mach. unavail. for 4-5 periods;
- **Plant Shutdown (PS):** All machines unavail. for 1-2 periods.

Disruption impact on nominal solution:

By repairing $\mathcal{S}^o$ into $\mathcal{S}^r$:

- Only ~4% of setup variables are modified;
- But total cost doubles on average (+101% cost increase), due to massive lost sales penalties.

Nominal solutions and disruptions

Nominal solutions:

Solving the LSP MILP for a long time budget (6000 wu).

Disruptions:

- **Machine Breakdown (MB):** 1 mach. unavail. for 4-5 periods;
- **Plant Shutdown (PS):** All machines unavail. for 1-2 periods.

Disruption impact on nominal solution:

By repairing $\mathcal{S}^o$ into $\mathcal{S}^r$:

- Only ~4% of setup variables are modified;
- But total cost doubles on average (+101% cost increase), due to massive lost sales penalties.

Reoptimization model and GNN training

Neighborhood constraint parameters:

- $\mathcal{T} = 10$, number of periods in short-term horizon;
- $\mathcal{K} = 10$, maximum number of setup changes.

GNN training:

- **Labeled data** computation (6000wu);
- **Hyperparameter tuning** by grid-search (embedding dimension, number of convolution blocks, learning rate, and focal loss parameters).

[Lin et al., 2017] Focal loss for dense object detection

Reoptimization model and GNN training

Neighborhood constraint parameters:

- $\mathcal{T} = 10$, number of periods in short-term horizon;
- $\mathcal{K} = 10$, maximum number of setup changes.

GNN training:

- **Labeled data** computation (6000wu);
- **Hyperparameter tuning** by grid-search (embedding dimension, number of convolution blocks, learning rate, and focal loss parameters).

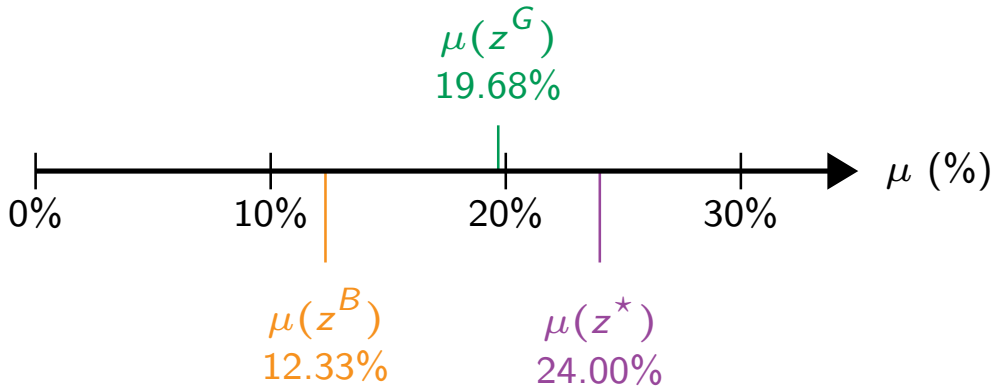
[Lin et al., 2017] Focal loss for dense object detection

Applying and comparing GNN and Baseline strategies

Improvement over repaired:

$$\mu(z) = \frac{z^r - z}{z^r}$$

computed on Baseline, GNN-aided and near-optimal solution values:



(with selection of $|\mathcal{Y}| = 40$ short-term setup variables to keep free)

Plan

Conclusion & Future work

Conclusion:

Proposed a novel GNN-aided fix-and-optimize strategy for fast MILP reoptimization of a Lot Sizing Problem:

- generalizing across varying instances and disruptions;
- outperforming significantly the Baseline strategy.

↪ Article about to be submitted.

Perspectives:

- Extend work to handle other disruptions (*e.g.* sudden demand peaks or order cancellations);
- Apply the framework to other optimization problems;
- Transition from a single-shot supervised prediction to a Reinforcement Learning (RL) agent guiding a sequential Large Neighborhood Search (LNS).

Conclusion & Future work

Conclusion:

Proposed a novel GNN-aided fix-and-optimize strategy for fast MILP reoptimization of a Lot Sizing Problem:

- generalizing across varying instances and disruptions;
- outperforming significantly the Baseline strategy.

↪ Article about to be submitted.

Perspectives:

- Extend work to handle other disruptions (*e.g.* sudden demand peaks or order cancellations);
- Apply the framework to other optimization problems;
- Transition from a single-shot supervised prediction to a Reinforcement Learning (RL) agent guiding a sequential Large Neighborhood Search (LNS).

References I

Gasse, M., Chételat, D., Ferroni, N., Charlin, L., and Lodi, A. (2019).
Exact Combinatorial Optimization with Graph Convolutional Neural Networks.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017).
Focal loss for dense object detection.
In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

References II

Lodi, A., Mossina, L., and Rachelson, E. (2020).
Learning to handle parameter perturbations in combinatorial optimization: An application to facility location.
EURO Journal on Transportation and Logistics, 9(4):100023.

Morabit, M., Desaulniers, G., and Lodi, A. (2024).
Learning to repeatedly solve routing problems.
Networks, 83(3):503-526.