ROADEF 2025

ML-guided MILP reoptimization applied to LSP

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

February 28, 2025

1 ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

2 DecisionBrain™ Smarter Decisions. Better Results.

Plan

Plan

Motivating example - Lot Sizing Problem (LSP)

Lot Sizing Problem (LSP):

Given:

- a planning horizon discretized into **periods**;
- a set of **machines** with limited capacities;
- a set of **items**, such that each item has:
  - initial inventory,
  - demands over time periods,
  - production unit and fixed setup resource consumption,
  - setup, production, inventory and lost sales unitary costs;

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

Motivating example - Lot Sizing Problem (LSP)

Lot Sizing Problem (LSP):

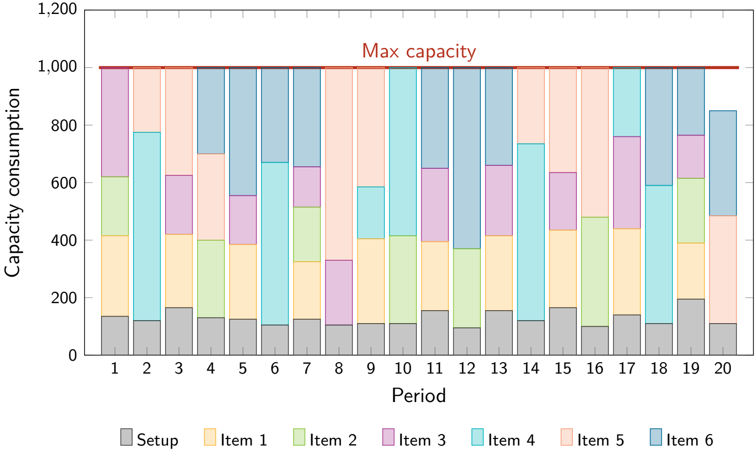
Given:

- a planning horizon discretized into **periods**;
- a set of **machines** with limited capacities;
- a set of **items**, such that each item has:
    - initial inventory,
    - demands over time periods,
    - production unit and fixed setup resource consumption,
    - setup, production, inventory and lost sales unitary costs;

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

Motivating example - LSP solution

Productions on one machine:



Motivating example - LSP MILP model

LSP MILP model:

min setup + production + inventory + lost sales costs

s.t. production and inventory vs demand and lost sales constraints

capacity constraints

minimum production constraints

[...]

$Y_{mit} \in \{0,1\}$ $m \in \{\text{machines}\},\ i \in \{\text{items}\},\ t \in \{\text{periods}\}$

$X_{mit} \geq 0$ $m \in \{\text{machines}\},\ i \in \{\text{items}\},\ t \in \{\text{periods}\}$

[...]

Motivating example - LSP MILP model

LSP MILP model:

min setup + production + inventory + lost sales costs

s.t. production and inventory vs demand and lost sales constraints

capacity constraints

minimum production constraints

[...]

$Y_{mit} \in \{0,1\}$ $m \in \{\text{machines}\},\ i \in \{\text{items}\},\ t \in \{\text{periods}\}$

$X_{mit} \geq 0$ $m \in \{\text{machines}\},\ i \in \{\text{items}\},\ t \in \{\text{periods}\}$

[...]

Motivating example - LSP MILP model

LSP MILP model:

min setup + production + inventory + lost sales costs

s.t. production and inventory vs demand and lost sales constraints

capacity constraints

minimum production constraints

[...]

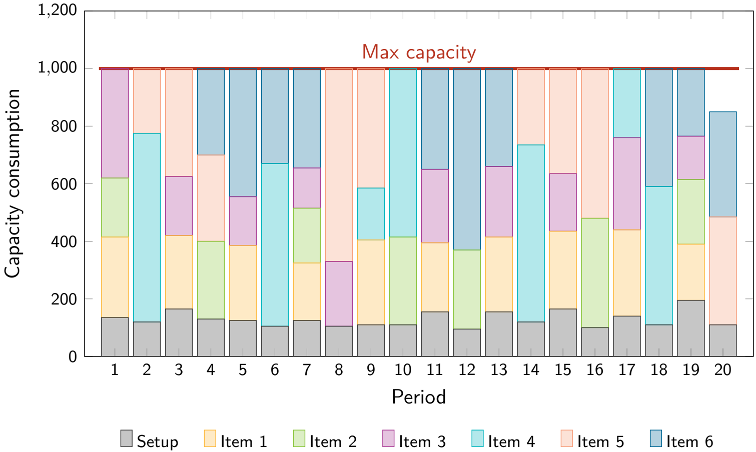
$Y_{mit} \in \{0,1\} \quad m \in \{\text{machines}\},\ i \in \{\text{items}\},\ t \in \{\text{periods}\}$

$X_{mit} \geq 0 \quad m \in \{\text{machines}\},\ i \in \{\text{items}\},\ t \in \{\text{periods}\}$

[...]

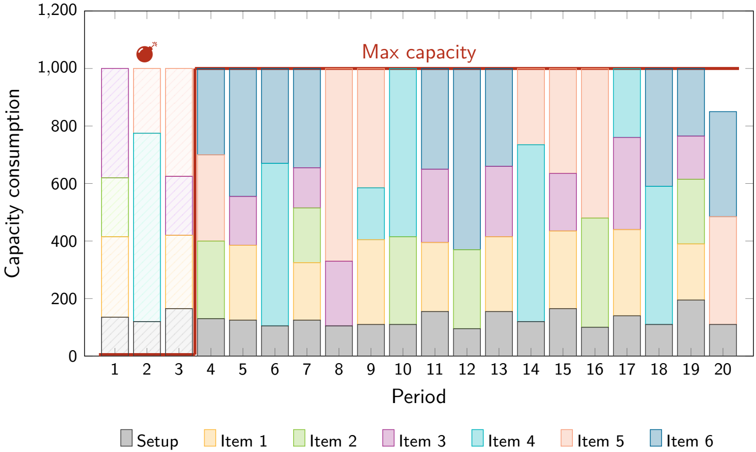
Motivating example - LSP perturbation

Productions perturbed due to machine breakdown:



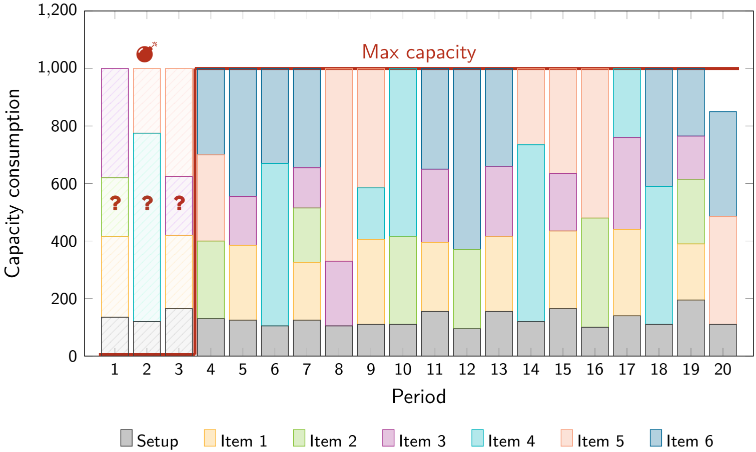
Motivating example - LSP perturbation

Productions perturbed due to machine breakdown:



Motivating example - LSP perturbation

Productions perturbed due to machine breakdown:



Plan

General context - Original setting

Original setting:

- NP-hard combinatorial optimization problem (*e.g.* LSP) modeled as a **MILP** Π;
- $\mathcal{I}$ **instance**;
- $\mathcal{S}$ (optimal or near-optimal) **solution** of $\mathcal{I}$:
  - obtained after a long computation time (*e.g.* hours),
  - using an MILP solver.

General context - Perturbed setting

Perturbed setting:

A short time before the execution of $\mathcal{S}$:

- **Perturbations** $\mathcal{P}$ are observed (*e.g.* machine breakdown),
  - affecting the coefficients of $\mathcal{I}$ (*e.g.* capacity coefficient),
  - and invalidating $\mathcal{S}$ (w.r.t. feasibility or optimality).
- A new instance, "**perturbed instance**", $\mathcal{I}'$ can be defined:
  - with the same dimensions as $\mathcal{I}$,
  - but with coefficients slightly different.

General context - Perturbed setting

Perturbed setting:

A short time before the execution of $\mathcal{S}$:

- **Perturbations** $\mathcal{P}$ are observed (*e.g.* machine breakdown),
  - affecting the coefficients of $\mathcal{I}$ (*e.g.* capacity coefficient),
  - and invalidating $\mathcal{S}$ (w.r.t. feasibility or optimality).
- A new instance, "**perturbed instance**", $\mathcal{I}'$ can be defined:
  - with the same dimensions as $\mathcal{I}$,
  - but with coefficients slightly different.

General context - Needs

Needs:

Finding a **new solution** $\mathcal{S}'$ while satisfying various criteria:

(a) adaptation of $\mathcal{S}'$ to perturbations;

(b) good quality of $\mathcal{S}'$;

(c) short computation time (*e.g.* a few tens of seconds or minutes);

(d) controlled deviation of $\mathcal{S}'$ from original solution $\mathcal{S}$.

? How to compute such an $\mathcal{S}'$?

- using an MILP-based approach;
- involving Machine Learning (ML).

General context - Needs

Needs:

Finding a **new solution** $\mathcal{S}'$ while satisfying various criteria:

(a) adaptation of $\mathcal{S}'$ to perturbations;

(b) good quality of $\mathcal{S}'$;

(c) short computation time (*e.g.* a few tens of seconds or minutes);

(d) controlled deviation of $\mathcal{S}'$ from original solution $\mathcal{S}$.

? How to compute such an $\mathcal{S}'$?

- using an MILP-based approach;
- involving Machine Learning (ML).

Plan

Related works - Reoptimization of NP-hard problems

Reoptimization of NP-hard problems:

Several works on reoptimizing NP-hard problems, such as works on:

- Scheduling Problems, *e.g.* [Schäffter, 1997];
- Traveling Sales Problems, *e.g.* [Archetti et al., 2003];
- Steiner Tree Problems, *e.g.* [Böckenhauer et al., 2008].

⚠ the methods can only be applied to these **specific problems** and assume quite **restrictive instance perturbations**.

[Schäffter, 1997] Scheduling with forbidden sets
[Archetti et al., 2003] Reoptimizing the Traveling Salesman Problem
[Böckenhauer et al., 2008] On the hardness of reoptimization

Related works - Reoptimization of NP-hard problems

Reoptimization of NP-hard problems:

Several works on reoptimizing NP-hard problems, such as works on:

- Scheduling Problems, *e.g.* [Schäffter, 1997];
- Traveling Sales Problems, *e.g.* [Archetti et al., 2003];
- Steiner Tree Problems, *e.g.* [Böckenhauer et al., 2008].

⚠ the methods can only be applied to these **specific problems** and assume quite **restrictive instance perturbations**.

[Schäffter, 1997] Scheduling with forbidden sets
[Archetti et al., 2003] Reoptimizing the Traveling Salesman Problem
[Böckenhauer et al., 2008] On the hardness of reoptimization

Related works - ML and MILP reoptimization

ML and MILP reoptimization:

Several works on MILP reoptimization leveraging ML, among them:

- [Xavier et al., 2021]
  - ML for initializing a separation-like algorithm;
  - limited to problems solvable through separation techniques.
- [Lodi et al., 2020] and [Morabit et al., 2023]
  - ML for defining a reoptimization problem whose feasible solution space is reduced compared to the original one;
  - require training an ML model for each instance dimension.

[Xavier et al., 2021] Learning to solve large-scale security-constrained Unit Commitment Problems
[Lodi et al., 2020] Learning to handle parameter perturbations in Comb. Opt.: an application to Facility Location
[Morabit et al., 2023] Learning to repeatedly solve routing problems

Related works - ML and MILP reoptimization

ML and MILP reoptimization:

Several works on MILP reoptimization leveraging ML, among them:

- [Xavier et al., 2021]
  - ML for initializing a separation-like algorithm;
  - limited to problems solvable through separation techniques.
- [Lodi et al., 2020] and [Morabit et al., 2023]
  - ML for defining a reoptimization problem whose feasible solution space is reduced compared to the original one;
  - require training an ML model for each instance dimension.

[Xavier et al., 2021] Learning to solve large-scale security-constrained Unit Commitment Problems
[Lodi et al., 2020] Learning to handle parameter perturbations in Comb. Opt.: an application to Facility Location
[Morabit et al., 2023] Learning to repeatedly solve routing problems

Related works - Our ambition

Our ambition in relation to existing works:

Designing an MILP-based approach, leveraging ML techniques, for reoptimizing solutions after instance perturbations, which:

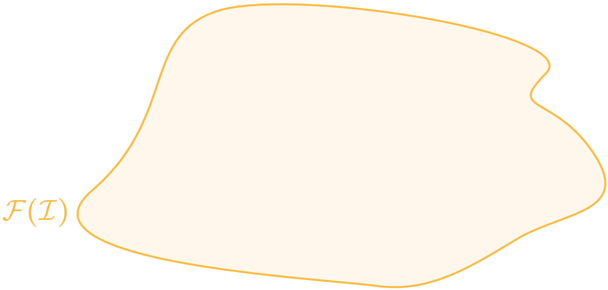
- considers “complex” perturbations;
- is applicable to various problems;
- handles instances of various dimensions.

Plan

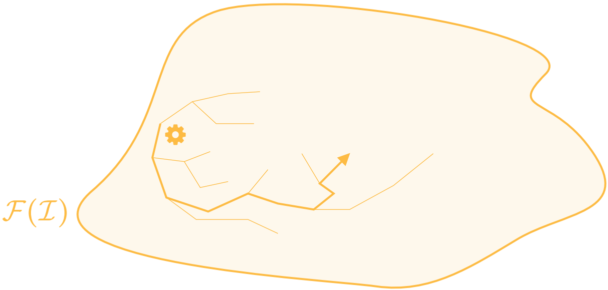
Plan

1. Introduction
2. Related works
3. Our reoptimization approach
   - Towards ML-guided MILP reoptimization
   - Parametric neighborhood
   - Framework
4. Use of Graph Convolutional Neural Networks (GCNNs)
5. Conclusion

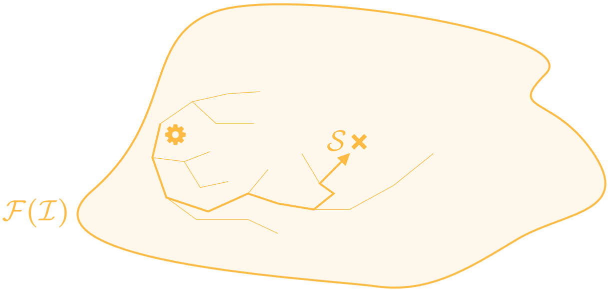
Towards ML-guided MILP reopt. - Original setting



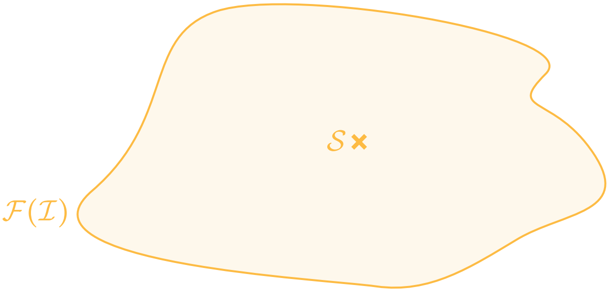
Towards ML-guided MILP reopt. - Original setting



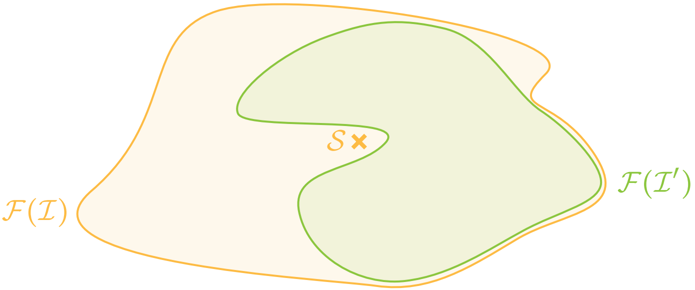
Towards ML-guided MILP reopt. - Original setting



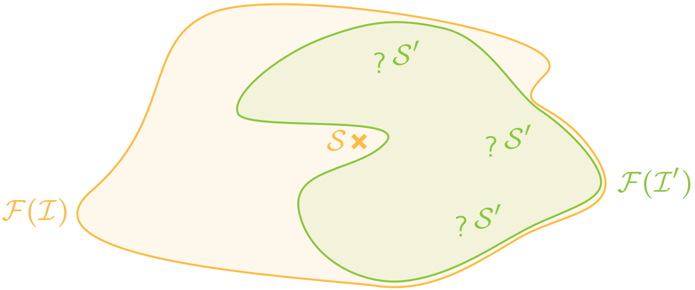
Towards ML-guided MILP reopt. - Original setting



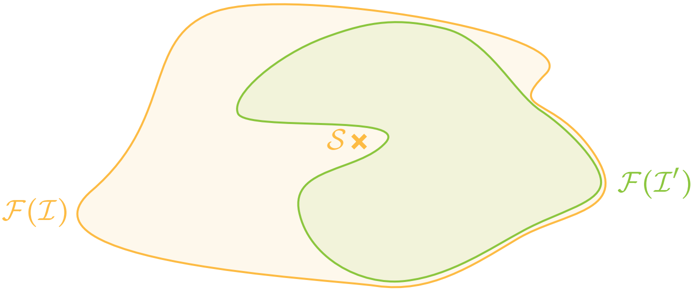
Towards ML-guided MILP reopt. - Perturbed setting



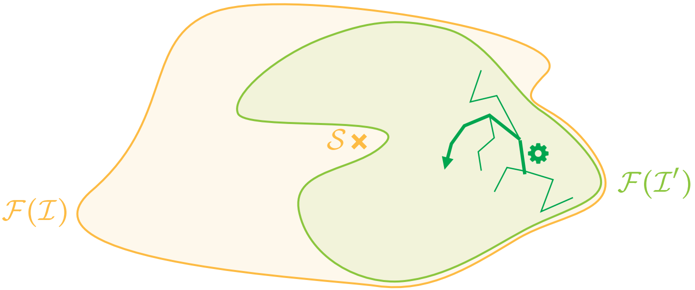
Towards ML-guided MILP reopt. - Perturbed setting



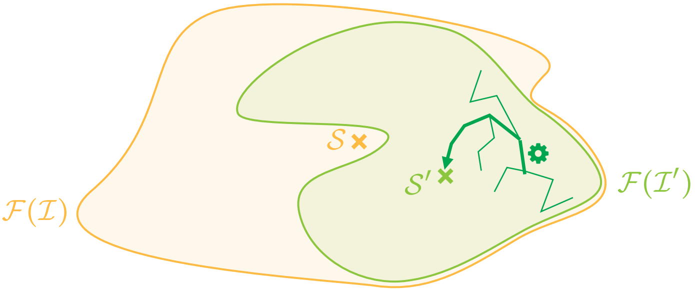
Towards ML-guided MILP reopt. - Naive approach



Towards ML-guided MILP reopt. - Naive approach



Towards ML-guided MILP reopt. - Naive approach



Towards ML-guided MILP reopt. - Naive approach

Naive approach:

Obtain $\mathcal{S}'$ by solving original MILP Π, on new instance $\mathcal{I}'$.

So that:

(a)✓ $\mathcal{S}'$ is feasible w.r.t. $\mathcal{I}'$;

but:

(b)(c)✗ computing a "good" $\mathcal{S}'$ is likely to take a long time;

(d)✗ $\mathcal{S}'$ is free to deviate indefinitely from $\mathcal{S}$.

Towards ML-guided MILP reopt. - Naive approach

Naive approach:

Obtain $\mathcal{S}'$ by solving original MILP Π, on new instance $\mathcal{I}'$.

So that:

(a)✓ $\mathcal{S}'$ is feasible w.r.t. $\mathcal{I}'$;

but:

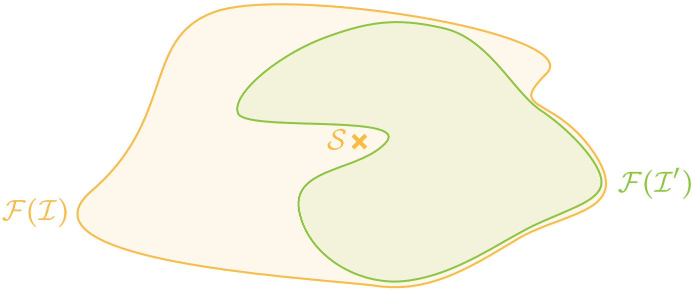
(b)(c)✗ computing a "good" $\mathcal{S}'$ is likely to take a long time;

(d)✗ $\mathcal{S}'$ is free to deviate indefinitely from $\mathcal{S}$.

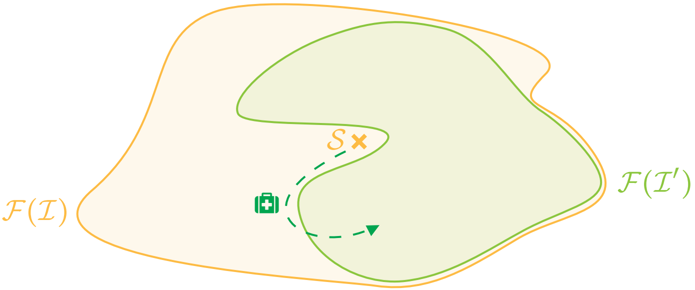
Towards ML-guided MILP reopt. - First assumption

First assumption:
We assume that we know a **repairing method** with which, from $\mathcal{S}$, we can build $\mathcal{S}'_r$ a new solution feasible w.r.t. $\mathcal{I}'$.

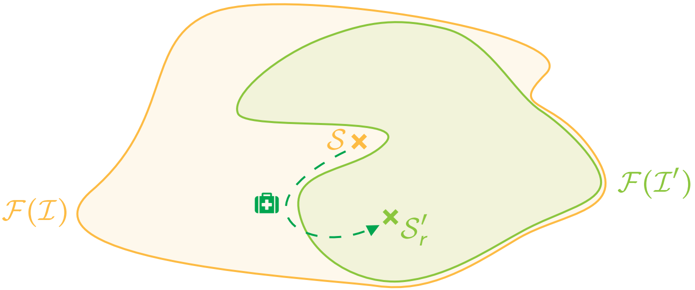
Towards ML-guided MILP reopt. - Baseline approach



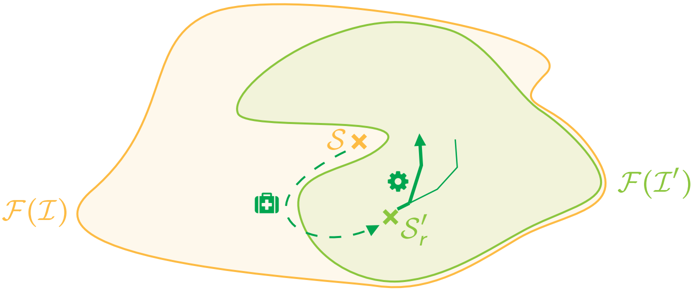
Towards ML-guided MILP reopt. - Baseline approach



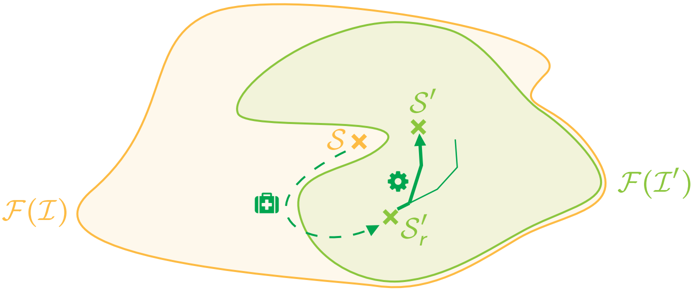
Towards ML-guided MILP reopt. - Baseline approach



Towards ML-guided MILP reopt. - Baseline approach



Towards ML-guided MILP reopt. - Baseline approach



Towards ML-guided MILP reopt. - Baseline approach

Baseline approach:

Obtain $\mathcal{S}'$ by solving original MILP Π, on new instance $\mathcal{I}'$, and warm-started with $\mathcal{S}'_r$.

So that:

(a)✓ $\mathcal{S}'$ is feasible w.r.t. $\mathcal{I}'$;

but:

(b)(c)≈ computing a "good" $\mathcal{S}'$ may still take a long time;

(d)✗ $\mathcal{S}'$ is still quite free to deviate indefinitely from $\mathcal{S}$.

Towards ML-guided MILP reopt. - Baseline approach

Baseline approach:

Obtain $\mathcal{S}'$ by solving original MILP Π, on new instance $\mathcal{I}'$, and warm-started with $\mathcal{S}'_r$.

So that:

(a)✓ $\mathcal{S}'$ is feasible w.r.t. $\mathcal{I}'$;

but:

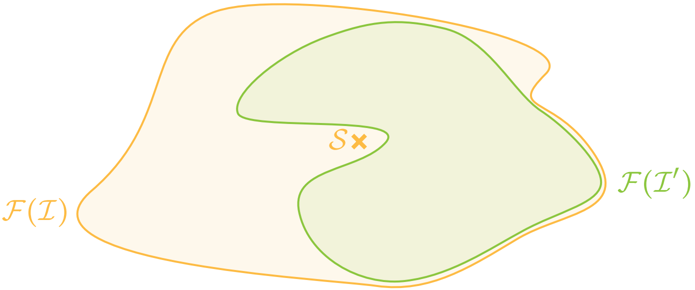
(b)(c)$^{\approx}$ computing a "good" $\mathcal{S}'$ may still take a long time;

(d)✗ $\mathcal{S}'$ is still quite free to deviate indefinitely from $\mathcal{S}$.

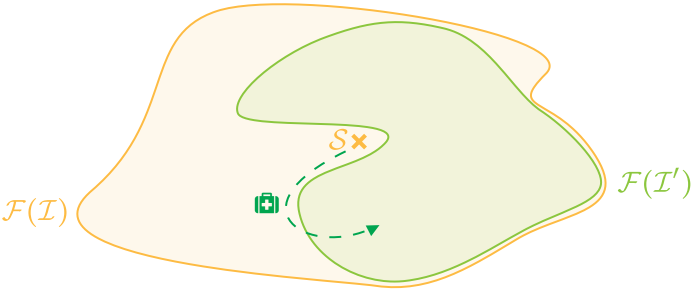
Towards ML-guided MILP reopt. - Second assumption

Second assumption:
We assume that we can define $\mathcal{N}(\mathcal{S})$ a **neighborhood** around $\mathcal{S}$, which contains the repaired solution $\mathcal{S}'_r$.

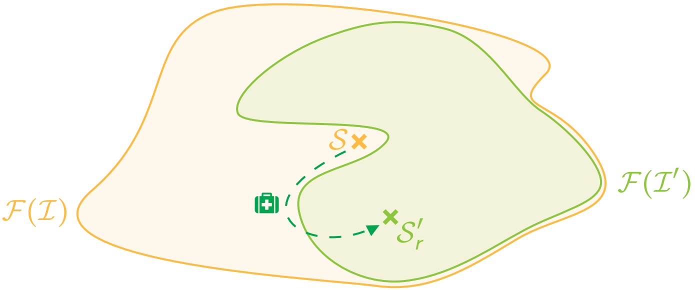
Towards ML-guided MILP reopt. - Local reoptimization



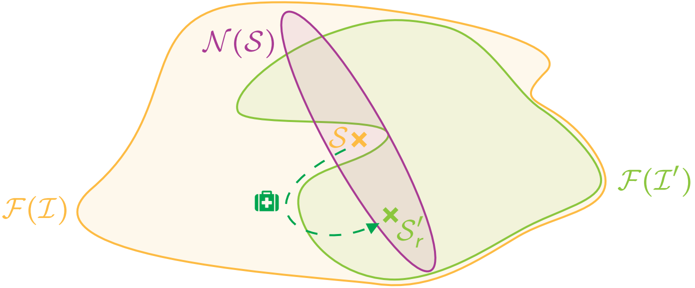
Towards ML-guided MILP reopt. - Local reoptimization



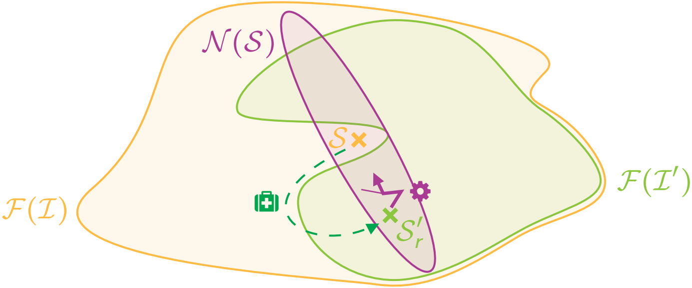
Towards ML-guided MILP reopt. - Local reoptimization



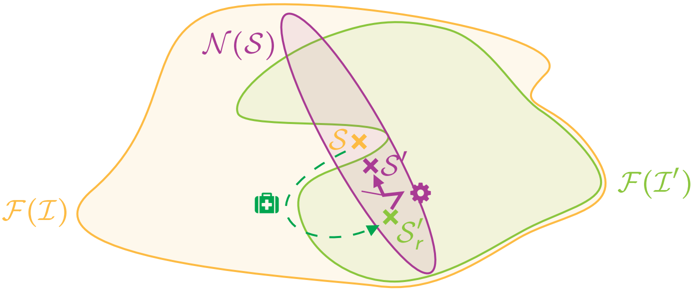
Towards ML-guided MILP reopt. - Local reoptimization



Towards ML-guided MILP reopt. - Local reoptimization



Towards ML-guided MILP reopt. - Local reoptimization



Towards ML-guided MILP reopt. - Local reoptimization

Local reoptimization:
Obtain $\mathcal{S}'$ by solving a new MILP $\Pi_{\mathcal{N}(\mathcal{S})}$, which:

- is built on the original MILP Π;
- has constraints enforcing $\mathcal{S}'$ to be in neighborhood $\mathcal{N}(\mathcal{S})$;
- and is warm-started with $\mathcal{S}'_r$.

So that:

(a)✓ $\mathcal{S}'$ is feasible w.r.t. $\mathcal{I}'$;

(b)(c)✓ computing a "good" $\mathcal{S}'$ might be more efficient, as the solution space is smaller;

(d)✓ the deviation between $\mathcal{S}$ and $\mathcal{S}'$ is controlled.

Use **Machine Learning (ML)** to choose the neighborhood $\mathcal{N}(\mathcal{S})$.

Towards ML-guided MILP reopt. - Local reoptimization

Local reoptimization:

Obtain $\mathcal{S}'$ by solving a new MILP $\Pi_{\mathcal{N}(\mathcal{S})}$, which:

- is built on the original MILP Π;
- has constraints enforcing $\mathcal{S}'$ to be in neighborhood $\mathcal{N}(\mathcal{S})$;
- and is warm-started with $\mathcal{S}'_r$.

So that:

(a)✓ $\mathcal{S}'$ is feasible w.r.t. $\mathcal{I}'$;

(b)(c)✓ computing a "good" $\mathcal{S}'$ might be more efficient, as the solution space is smaller;

(d)✓ the deviation between $\mathcal{S}$ and $\mathcal{S}'$ is controlled.

Use **Machine Learning (ML)** to choose the neighborhood $\mathcal{N}(\mathcal{S})$.

Towards ML-guided MILP reopt. - Local reoptimization

Local reoptimization:

Obtain $\mathcal{S}'$ by solving a new MILP $\Pi_{\mathcal{N}(\mathcal{S})}$, which:

- is built on the original MILP Π;
- has constraints enforcing $\mathcal{S}'$ to be in neighborhood $\mathcal{N}(\mathcal{S})$;
- and is warm-started with $\mathcal{S}'_r$.

So that:

(a)✓ $\mathcal{S}'$ is feasible w.r.t. $\mathcal{I}'$;

(b)(c)✓ computing a "good" $\mathcal{S}'$ might be more efficient, as the solution space is smaller;

(d)✓ the deviation between $\mathcal{S}$ and $\mathcal{S}'$ is controlled.

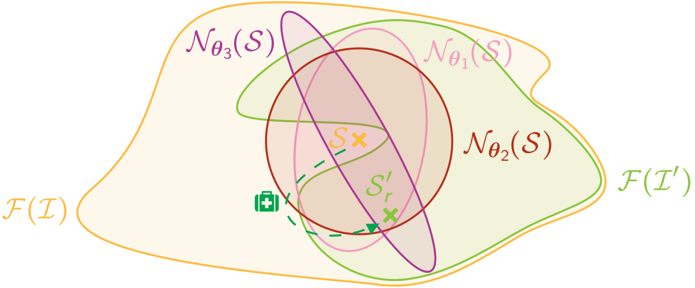
💡 Use **Machine Learning (ML)** to choose the neighborhood $\mathcal{N}(\mathcal{S})$.

Plan

Parametric neighborhood

Parametric neighborhood:
Use of a parametric neighborhood $\mathcal{N}_{\boldsymbol{\theta}}(\mathcal{S})$,
with $\boldsymbol{\theta} \in \mathbb{N}^K$ vector of parameters controlling its size
(where dimension K depends on the studied problem).

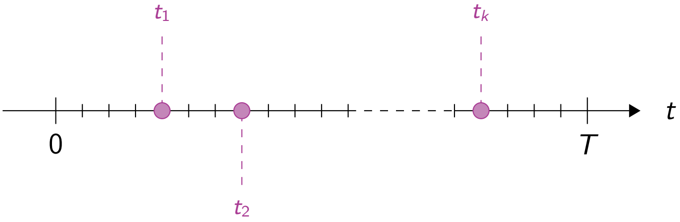
Parametric neighborhood



Parametric neighborhood - LSP neighborhood

Let $\mathcal{S}$ be a LSP solution, m a machine and i an item.

Setups of item i on machine m:



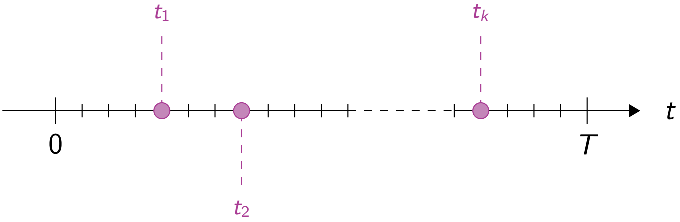
$\mathcal{S}$, as a MILP solution of Π, verifies:

- $Y^{\star}_{mit} = 1$, for $t \in \{t_1, t_2, \ldots, t_k\}$;
- $Y^{\star}_{mit} = 0$, otherwise.

Parametric neighborhood - LSP neighborhood

Let $\mathcal{S}$ be a LSP solution, m a machine and i an item.

Setups of item i on machine m:



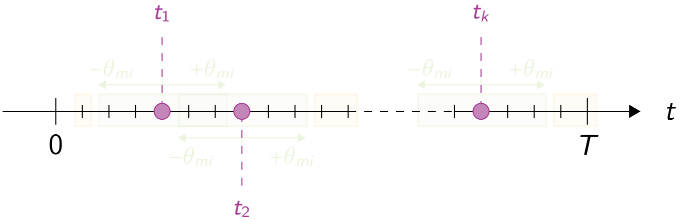
$\mathcal{S}$, as a MILP solution of Π, verifies:

- $Y^{\star}_{mit} = 1$, for $t \in \{t_1, t_2, \dots, t_k\}$;
- $Y^{\star}_{mit} = 0$, otherwise.

Parametric neighborhood - LSP neighborhood

Given $\mathcal{S}$, m and i, we choose $\theta_{mi} \in \mathbb{N}$.

Periods close to/far from setups:



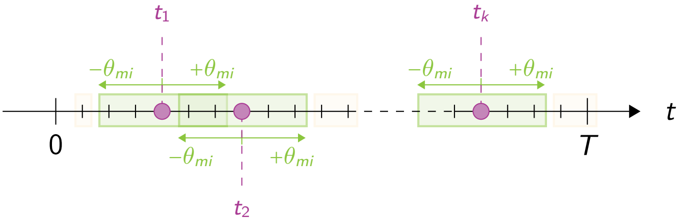
We call:

- $t \in$: periods close to setups of i on m;
- $t \in$: periods far from setups of i on m.

Parametric neighborhood - LSP neighborhood

Given $\mathcal{S}$, m and i, we choose $\theta_{mi} \in \mathbb{N}$.

Periods close to/far from setups:



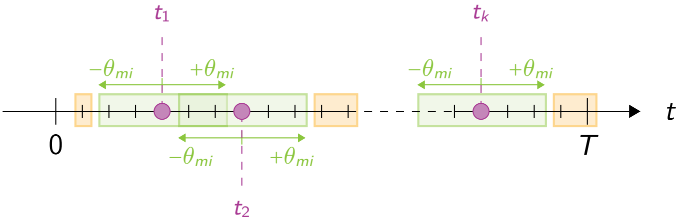
We call:

- $t \in$ [green box] : periods close to setups of i on m;
- $t \in$ [light box] : periods far from setups of i on m.

Parametric neighborhood - LSP neighborhood

Given $\mathcal{S}$, m and i, we choose $\theta_{mi} \in \mathbb{N}$.

Periods close to/far from setups:



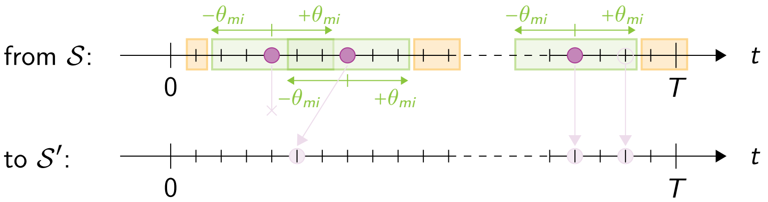
We call:

- $t \in$ [green box] : periods close to setups of i on m;
- $t \in$ [orange box] : periods far from setups of i on m.

Parametric neighborhood - LSP neighborhood

Given $\mathcal{S}$, we choose $\theta_{mi} \in \mathbb{N}$ for each machine m and item i.

Neighborhood of $\mathcal{S}$ - Allowed operations on setups:



with $\mathcal{S}' \in \mathcal{N}_{\boldsymbol{\theta}}(\mathcal{S})$.

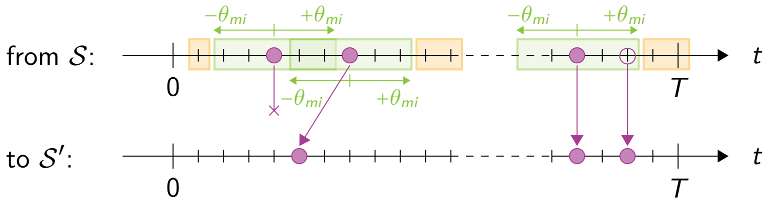
New MILP model $\Pi_{\mathcal{N}_\theta(\mathcal{S})}$ contains:

- $Y_{mit} = 0$, for $t \in$ ☐ (far from setups);
- $Y_{mit} \in \{0, 1\}$, for $t \in$ ☐ (close to setups).

Parametric neighborhood - LSP neighborhood

Given $\mathcal{S}$, we choose $\theta_{mi} \in \mathbb{N}$ for each machine m and item i.

Neighborhood of $\mathcal{S}$ - Allowed operations on setups:



with $\mathcal{S}' \in \mathcal{N}_{\boldsymbol{\theta}}(\mathcal{S})$.

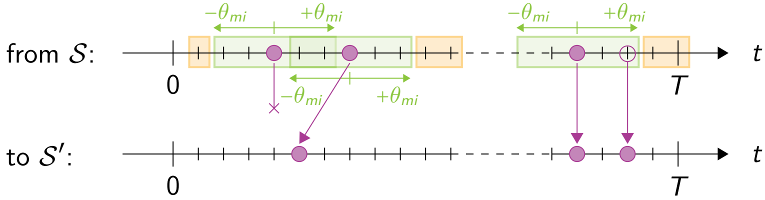
New MILP model $\Pi_{\mathcal{N}_\theta(\mathcal{S})}$ contains:

- $Y_{mit} = 0$, for $t \in$ ☐ (far from setups);
- $Y_{mit} \in \{0, 1\}$, for $t \in$ ☐ (close to setups).

Parametric neighborhood - LSP neighborhood

Given $\mathcal{S}$, we choose $\theta_{mi} \in \mathbb{N}$ for each machine m and item i.

Neighborhood of $\mathcal{S}$ - Allowed operations on setups:



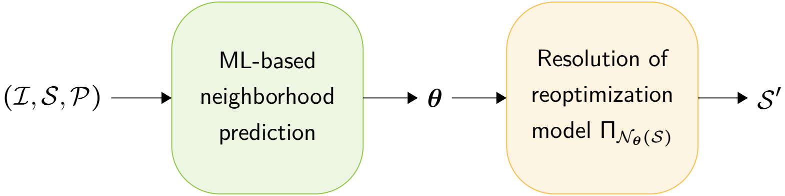
with $\mathcal{S}' \in \mathcal{N}_{\boldsymbol{\theta}}(\mathcal{S})$.

New MILP model $\Pi_{\mathcal{N}_{\boldsymbol{\theta}}(\mathcal{S})}$ contains:

- $Y_{mit} = 0$, for $t \in$ ☐ (far from setups);
- $Y_{mit} \in \{0, 1\}$, for $t \in$ ☐ (close to setups).

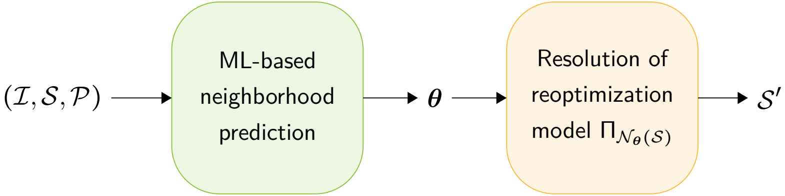
Plan

Framework



- What ML model to choose for predicting $\boldsymbol{\theta}$?
- Dimensions of the inputs $(\mathcal{I}, \mathcal{S}, \mathcal{P})$ may vary.

Framework



- What ML model to choose for predicting $\boldsymbol{\theta}$?
- Dimensions of the inputs $(\mathcal{I}, \mathcal{S}, \mathcal{P})$ may vary.

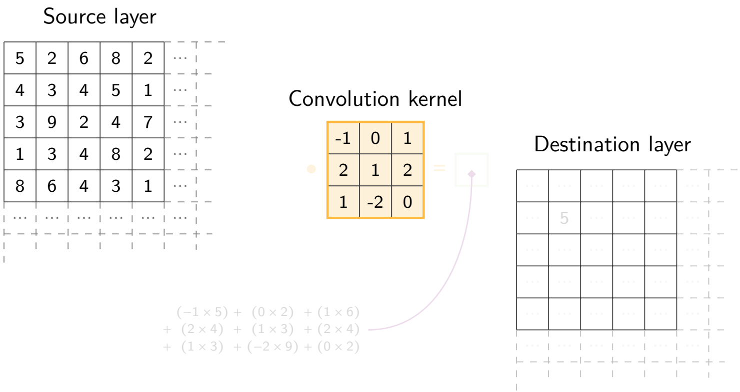
Plan

1. Introduction
2. Related works
3. Our reoptimization approach
4. Use of Graph Convolutional Neural Networks (GCNNs)
5. Conclusion

Plan

Brief introduction to GCNNs

Convolution with a kernel in CNNs:



Brief introduction to GCNNs

Convolution with a kernel in CNNs:

Source layer

5	2	6	8	2	···
4	3	4	5	1	···
3	9	2	4	7	···
1	3	4	8	2	···
8	6	4	3	1	···
···	···	···	···	···	···

Convolution kernel

-1	0	1
2	1	2
1	-2	0

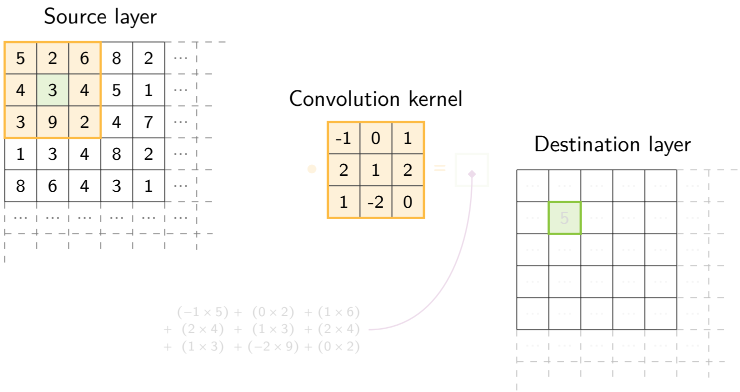
Destination layer

···	···	···	···	···	···
···	5	···	···	···	···
···	···	···	···	···	···
···	···	···	···	···	···
···	···	···	···	···	···
···	···	···	···	···	···

$$(-1 \times 5) + (0 \times 2) + (1 \times 6) + (2 \times 4) + (1 \times 3) + (2 \times 4) + (1 \times 3) + (-2 \times 9) + (0 \times 2)$$

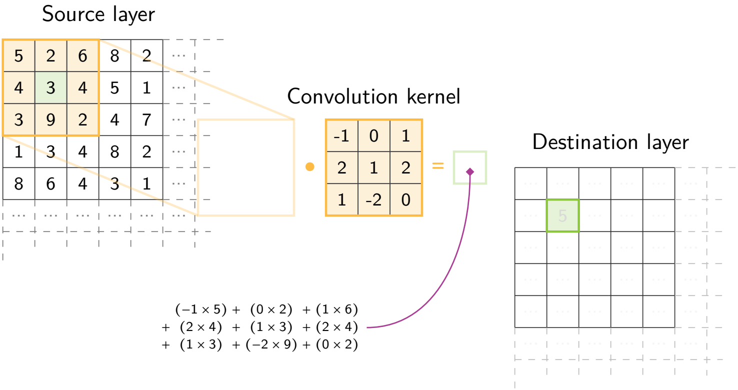
Brief introduction to GCNNs

Convolution with a kernel in CNNs:



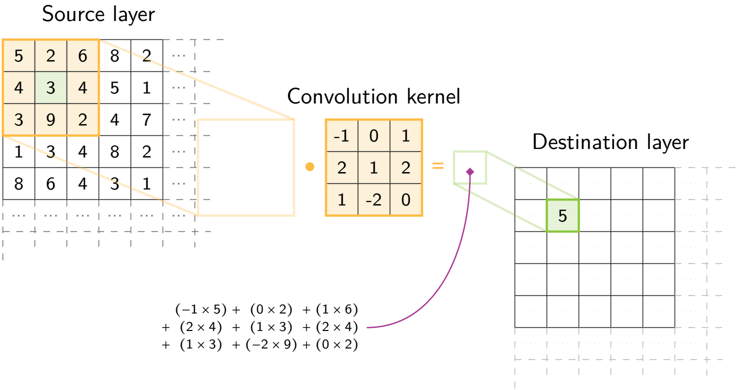
Brief introduction to GCNNs

Convolution with a kernel in CNNs:



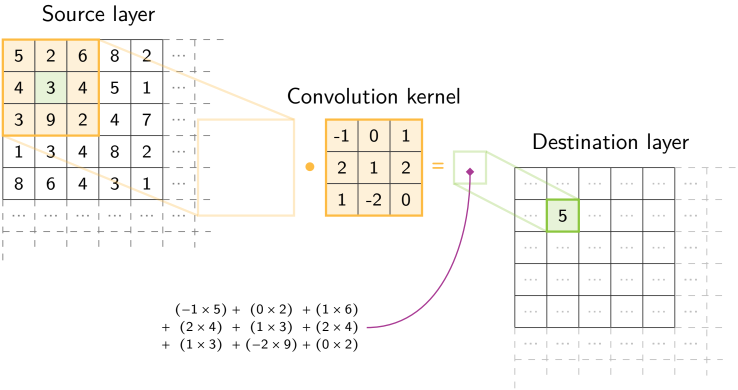
Brief introduction to GCNNs

Convolution with a kernel in CNNs:



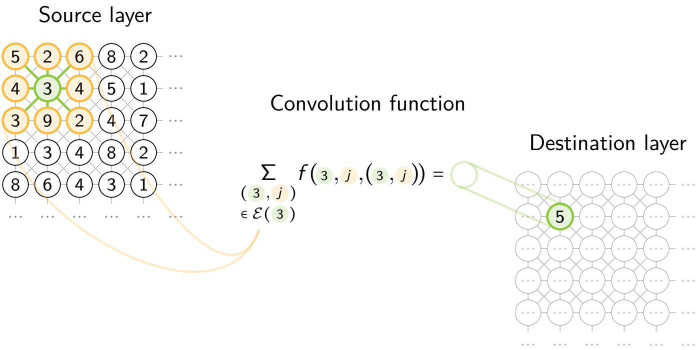
Brief introduction to GCNNs

Convolution with a kernel in CNNs:



Brief introduction to GCNNs

Graph interpretation of convolution with kernel:



Brief introduction to GCNNs

Essentially, GCNNs can be seen as **generalizations** of CNNs, where:

- **graphs**, with features associated to nodes and edges, are used instead of tensors;
- **convolution** is performed with a **function** instead of a kernel, such as:

$$v_i = f(v_i, \sum_{(i,j)\in\mathcal{E}(i)} g(v_i, v_j, e_{ij}))$$

with v_i (resp. e_{ij}) feature vector of node (resp. edge).

Brief introduction to GCNNs

Essentially, GCNNs can be seen as **generalizations** of CNNs, where:

- **graphs**, with features associated to nodes and edges, are used instead of tensors;
- **convolution** is performed with a **function** instead of a kernel, such as:

$$v_i = f\left(v_i, \sum_{(i,j)\in\mathcal{E}(i)} g(v_i, v_j, e_{ij})\right)$$

with v_i (resp. e_{ij}) feature vector of node (resp. edge).

Brief introduction to GCNNs

Why using a GCNN?

In ML litterature (*e.g.* [Gasse et al., 2019]), GCNNs are known for:

- Being well-defined no matter the **input dimensions**;

↪ It will be useful as our inputs have various dimensions.

- Being adapted to **sparse** graphs;

↪ It will be useful as the graphs we use are sparse.

? How do we use GCNNs?

[Gasse et al., 2019] Exact combinatorial optimization with graph convolutional neural networks.

Brief introduction to GCNNs

Why using a GCNN?

In ML litterature (*e.g.* [Gasse et al., 2019]), GCNNs are known for:

- Being well-defined no matter the **input dimensions**;

↪ It will be useful as our inputs have various dimensions.

- Being adapted to **sparse** graphs;

↪ It will be useful as the graphs we use are sparse.

? How do we use GCNNs?

[Gasse et al., 2019] Exact combinatorial optimization with graph convolutional neural networks.

Brief introduction to GCNNs

Why using a GCNN?

In ML litterature (*e.g.* [Gasse et al., 2019]), GCNNs are known for:

- Being well-defined no matter the **input dimensions**;

↪ It will be useful as our inputs have various dimensions.

- Being adapted to **sparse** graphs;

↪ It will be useful as the graphs we use are sparse.

? How do we use GCNNs?

[Gasse et al., 2019] Exact combinatorial optimization with graph convolutional neural networks.

Brief introduction to GCNNs

Why using a GCNN?

In ML litterature (*e.g.* [Gasse et al., 2019]), GCNNs are known for:

- Being well-defined no matter the **input dimensions**;
- ↪ It will be useful as our inputs have various dimensions.
- Being adapted to **sparse** graphs;
- ↪ It will be useful as the graphs we use are sparse.

? How do we use GCNNs?

[Gasse et al., 2019] Exact combinatorial optimization with graph convolutional neural networks.

Plan

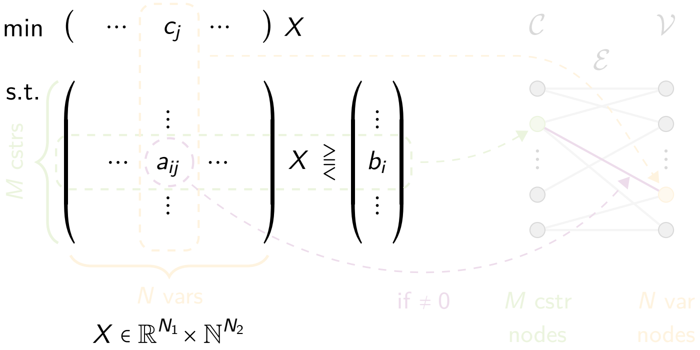
Embedding a MILP into a graph

Following [Gasse et al., 2019], we map a MILP into a **bipartite graph of features,** to represent $(\mathcal{I}, \mathcal{S}, \mathcal{P})$, as follows:

[Gasse et al., 2019] Exact combinatorial optimization with graph convolutional neural networks

Embedding a MILP into a graph

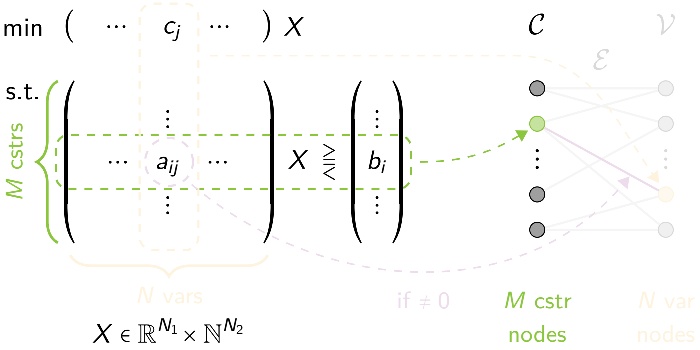
Following [Gasse et al., 2019], we map a MILP into a **bipartite graph of features,** to represent $(\mathcal{I}, \mathcal{S}, \mathcal{P})$, as follows:



[Gasse et al., 2019] Exact combinatorial optimization with graph convolutional neural networks

Embedding a MILP into a graph

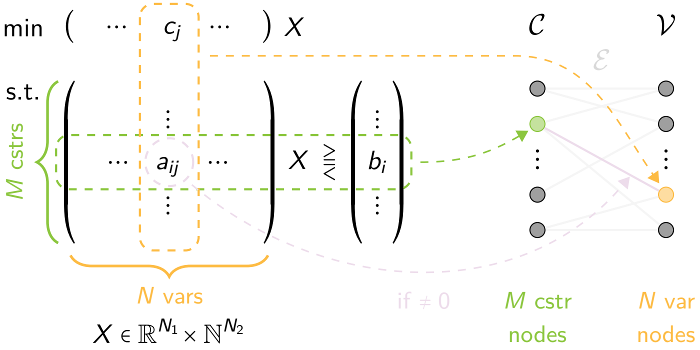
Following [Gasse et al., 2019], we map a MILP into a **bipartite graph of features,** to represent $(\mathcal{I}, \mathcal{S}, \mathcal{P})$, as follows:



[Gasse et al., 2019] Exact combinatorial optimization with graph convolutional neural networks

Embedding a MILP into a graph

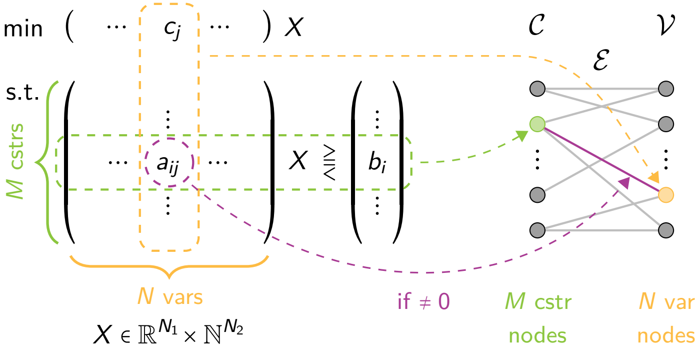
Following [Gasse et al., 2019], we map a MILP into a **bipartite graph of features,** to represent $(\mathcal{I}, \mathcal{S}, \mathcal{P})$, as follows:



[Gasse et al., 2019] Exact combinatorial optimization with graph convolutional neural networks

Embedding a MILP into a graph

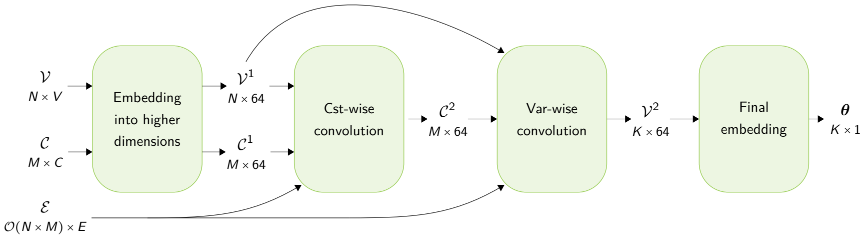
Following [Gasse et al., 2019], we map a MILP into a **bipartite graph of features,** to represent $(\mathcal{I}, \mathcal{S}, \mathcal{P})$, as follows:



[Gasse et al., 2019] Exact combinatorial optimization with graph convolutional neural networks

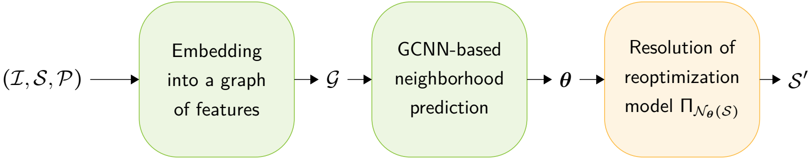
Plan

GCNN architecture



Plan

Framework overview with GCNN



Plan

Conclusion, Work in progress & Open questions

Conclusion:

Design of an MILP-based approach, leveraging GCNN techniques, for reoptimizing solutions after instance perturbations.

Work in progress & Open questions:

- We have performed some data collections (given $\mathcal{I}$, computation of $\mathcal{S}$, simulation of $\mathcal{P}$, ...).
  - ? What is a good θ? What is a good $\mathcal{N}_{\boldsymbol{\theta}}(\mathcal{S})$?
- We are currently implementing the GCNN model.
  - ? Inputs: What are relevant graph features to consider?
  - ? Outputs: How to have $\boldsymbol{\theta} \in \mathbb{N}^K$?
- Can we easily applied our approach to other problems?

Conclusion, Work in progress & Open questions

Conclusion:

Design of an MILP-based approach, leveraging GCNN techniques, for reoptimizing solutions after instance perturbations.

Work in progress & Open questions:

- We have performed some data collections (given $\mathcal{I}$, computation of $\mathcal{S}$, simulation of $\mathcal{P}$, ...).
  - ? What is a good $\boldsymbol{\theta}$? What is a good $\mathcal{N}_{\boldsymbol{\theta}}(\mathcal{S})$?
- We are currently implementing the GCNN model.
  - ? Inputs: What are relevant graph features to consider?
  - ? Outputs: How to have $\boldsymbol{\theta} \in \mathbb{N}^K$?
- Can we easily applied our approach to other problems?

Conclusion, Work in progress & Open questions

Conclusion:

Design of an MILP-based approach, leveraging GCNN techniques, for reoptimizing solutions after instance perturbations.

Work in progress & Open questions:

- We have performed some data collections (given $\mathcal{I}$, computation of $\mathcal{S}$, simulation of $\mathcal{P}$, ...).
  - ? What is a good $\boldsymbol{\theta}$? What is a good $\mathcal{N}_{\boldsymbol{\theta}}(\mathcal{S})$?
- We are currently implementing the GCNN model.
  - ? Inputs: What are relevant graph features to consider?
  - ? Outputs: How to have $\boldsymbol{\theta} \in \mathbb{N}^K$?
- Can we easily applied our approach to other problems?

Conclusion, Work in progress & Open questions

Conclusion:

Design of an MILP-based approach, leveraging GCNN techniques, for reoptimizing solutions after instance perturbations.

Work in progress & Open questions:

- We have performed some data collections (given $\mathcal{I}$, computation of $\mathcal{S}$, simulation of $\mathcal{P}$, ...).
  - ? What is a good $\boldsymbol{\theta}$? What is a good $\mathcal{N}_{\boldsymbol{\theta}}(\mathcal{S})$?
- We are currently implementing the GCNN model.
  - ? Inputs: What are relevant graph features to consider?
  - ? Outputs: How to have $\boldsymbol{\theta} \in \mathbb{N}^K$?
- Can we easily applied our approach to other problems?

References I

Archetti, C., Bertazzi, L., and Speranza, M. G. (2003).
Reoptimizing the Traveling Salesman Problem.
Networks, 42(3):154–159.

Böckenhauer, H.-J., Hromkovič, J., Mömke, T., and Widmayer, P. (2008).
On the hardness of reoptimization.
In Geffert, V., Karhumäki, J., Bertoni, A., Preneel, B., Návrat, P., and Bieliková, M., editors, *SOFSEM 2008: Theory and Practice of Computer Science*, pages 50–65, Berlin, Heidelberg. Springer Berlin Heidelberg.

References II

Gasse, M., Chételat, D., Ferroni, N., Charlin, L., and Lodi, A. (2019).
Exact combinatorial optimization with graph convolutional neural networks.

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. (2023).
Learning to repeatedly solve routing problems.
Networks, n/a(n/a).

References III

Schäffter, M. W. (1997).
Scheduling with forbidden sets.
Discrete Applied Mathematics, 72(1):155–166.
Models and Algorithms for Planning and Scheduling Problems.

Xavier, A. S., Qiu, F., and Ahmed, S. (2021).
Learning to solve large-scale security-constrained Unit Commitment Problems.
INFORMS Journal on Computing, 33(2):739–756.

Embedding a MILP into a graph - Variable features

Variable features:

To each variable node in $\mathcal{V}$, with corresponding MILP variable X, we associate $F_v = 7$ features related to the model and $(\mathcal{I}, \mathcal{S}, \mathcal{P})$:

- `is_binary` $\in \{0, 1\}$: whether X is binary;
- `obj_coef` $\in [-1, 1]$: objective coefficient related to X (normalized w.r.t. largest absolute objective coefficient);
- `has_lb` $\in \{0, 1\}$: whether X is bounded by a lb;
- `has_ub` $\in \{0, 1\}$: whether X is bounded by an ub;
- `sol_at_lb` $\in \{0, 1\}$: whether X value in $\mathcal{S}$ equals its lb;
- `sol_at_ub` $\in \{0, 1\}$: whether X value in $\mathcal{S}$ equals its ub;
- `sol_val` $\in [0, 1]$: X value in $\mathcal{S}$ (normalized).

Embedding a MILP into a graph - Constraint features

Constraint features:

To each constraint node in $\mathcal{C}$, we associate $F_c = 5$ features related to the model and $(\mathcal{I}, \mathcal{S}, \mathcal{P})$:

- `cos_sim` $\in [-1, 1]$: cosine similarity between constraint coefficients and objective coefficients;
- `is_equality` $\in \{0, 1\}$: whether the constraint is an equality one;
- `is_lower_inequality` $\in \{0, 1\}$: whether the constraint is a lower inequality one;
- `rhs` $\in [-1, 1]$: right-hand side (normalized w.r.t. largest constraint coefficient);
- `rhs_chg` $\in \mathbb{R}$: right-hand side change due to perturbations (normalized w.r.t. largest constraint coefficients before perturbations).

Embedding a MILP into a graph - Edge features

Edge features:

To each edge in $\mathcal{E}$, we associate $F_e = 1$ feature related to the model and $(\mathcal{I}, \mathcal{S}, \mathcal{P})$:

- `coef` $\in [-1, 1]$: coefficient (normalized w.r.t. largest constraint coefficient).

GCNN architecture - Convolutions

Constraint-wise and variable-wise convolutions:

$$\boldsymbol{c}_i \leftarrow \boldsymbol{f}_{cst}\left(\boldsymbol{c}_i, \sum_{j,\ (i,j)\in\mathcal{E}} \boldsymbol{g}_{cst}(\boldsymbol{c}_i,\ \boldsymbol{v}_j, \boldsymbol{e}_{i,j})\right),$$

$$\boldsymbol{v}_j \leftarrow \boldsymbol{f}_{var}\left(\boldsymbol{v}_j, \sum_{i,\ (i,j)\in\mathcal{E}} \boldsymbol{g}_{var}(\boldsymbol{c}_i,\ \boldsymbol{v}_j, \boldsymbol{e}_{i,j})\right).$$

where $\boldsymbol{f}_{cst}$, $\boldsymbol{f}_{var}$, $\boldsymbol{g}_{cst}$ and $\boldsymbol{g}_{var}$ $\simeq$ 2-layer perceptrons with relu activation functions.