Lecture group
MIP-GNN: A Data-Driven Framework for Guiding Combinatorial Solvers by *Khalil et al.*

Alexandre Schulz

September 26, 2022

Context

- enhancing MIP solvers with data-driven insights
- Graph Neural Networks (GNN)
- Predicting *variable biases*
- Application to Binary Linear Programs (BLP)
- Node selection and warm-starting

Variable biases

Let $I := (A, b, c) \in \mathbb{Q}^{m \times n} \times \mathbb{Q}^m \times \mathbb{Q}^n$ be an instance of a BLP problem. The set of integer solutions of I is :

$$F_{\mathrm{Int}}(I) = \{x \in \mathbb{Z}^n | Ax \leq b, x \in \mathbb{Z}^{*+}\} \tag{1}$$

The set of *near-optimal* solutions :

$$F_\epsilon^*(I) = \{x \in F_{\mathrm{Int}}(I) : |c^T x^* - c^T x| \leq \epsilon\} \tag{2}$$

Variable biases are defined as *[...] component-wise averaging over a set of near-optimal solutions [...]*

$$\bar{b}(I) = \frac{1}{|F_\epsilon^*|} \sum_{x \in F_\epsilon^*(I)} x \tag{3}$$

Training

Let $\mathcal{C}$ be a set of CO problems and $\mathcal{D}$ a distribution over $\mathcal{C}$. Training aims at learning a function $f_\theta : \mathcal{V} \to \mathbb{R}$ where the set of parameters is $\theta \in \Theta$. We note S the training set sampled from $\mathcal{D}$.

$$\min_{\theta \in \Theta} \frac{1}{|S|} \sum_{I \in S} l(f_\theta(\mathcal{V}(I)), \bar{b}(I)) \tag{4}$$

MIP-GNN architecture

- instance encoded as a bipartite graph $B(I) = (V(I), C(I), E(I))$
- variable-to-constraint (v-to-c)
- constraint-to-variable (c-to-v)

v-to-c and c-to-v layers are stacked in an alternating manner.

MIP-GNN architecture

Variable to constraint pass

Let

- $v_i^{(t)} \in \mathbb{R}^d$ variable features
- $c_j^{(t)} \in \mathbb{R}^d$ constraint features

Update the constraint embeddings:

$$c_j^{(t+1)} = f_{\text{merge}}^{W_{2,C}}\left(c_j^{(t)}, f_{\text{aggr}}^{W_{1,C}}(\{[v_i^{(t)}, A_{ji}, b_j] | v_i \in N(c_j)\})\right) \tag{5}$$

- $f_{\text{aggr}}^{W_{1,C}}$ aggregates over adjacent variables
- $f_{\text{merge}}^{W_{1,C}}$ merges constraint embeddings

MIP-GNN architecture

Constraint to variable pass

Assign a scalar value to the variables of the problem:

$$\bar{x}_i = f_{\text{asg}}^{W_a}(x_i^{(t)}) \tag{6}$$

Compute the *error message* : *[...] indicating how much the j-th constraint, [...], contributes to the constraints' violation in total.*

$$e = \text{softmax}(A\bar{x} - b) \in \mathbb{R}^m \tag{7}$$

Update the variable embeddings :

$$v_i^{(t+1)} = f_{\text{merge}}^{W_{2,V}}(v_i^{(t)}, f_{\text{aggr}}^{W_{1,V}}(\{[c_j^{(t)}, A_{ji}, b_j, e_j] | c_j \in N(v_i)\})) \tag{8}$$

Training

A this point predicting the variable assignments is a regression problem. Training can be simplified by transforming the problem into a *classification* problem : choose a threshold value $\tau > 0$ and assign classes :

$$\hat{b}_i = \begin{cases} 0 & \text{if} \quad \bar{b}_i \leq \tau \\ 1 & \text{otherwise} \end{cases} \tag{9}$$

Evaluation

Node selection

Let $\hat{p} \in [0,1]^n$ be the prediction of the model. Define a *confidence score* :

$$\text{score}(\hat{p}_i) = 1 - |\hat{p}_i - \lfloor \hat{p}_i \rceil| \tag{10}$$

where $\lfloor \cdot \rceil$ rounds to the nearest integer.
Define a *node score* used to guide the branching process as the sum of confidence scores (or complement) for the set of variables that are fixed the the current node :

$$\text{node-score}(N; \hat{p}) = \begin{cases} \text{score}(\hat{p}_i) & \text{if} x_i^N = \lfloor \hat{p}_i \rceil \\ 1 - \text{score}(\hat{p}_i) & \text{otherwise} \end{cases} \tag{11}$$

Evaluation

Node selection

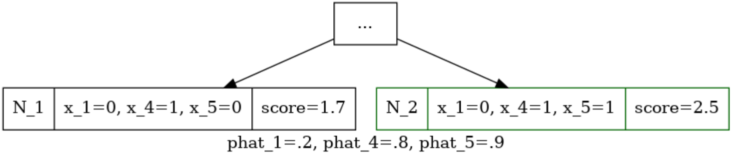


Figure: Node selection example

Evaluation

Warm starting

[...] attempt to directly construct a feasible solution via rounding. Introduction of a *rounding threshold* $p_{\text{min}} \in [0.5, 1)$. Variable bias prediction are rounded to the nearest integer :

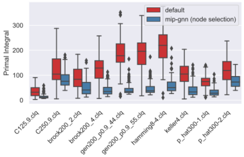
$$\hat{x}_i = \lfloor \hat{p}_i \rceil \quad \text{if} \quad \text{score}(\hat{p}_i) \geq p_{\text{min}} \tag{12}$$

Threshold grid values $\{.99, .98, .96, .92, .84, .68\}$, CPLEX's solution repair used to produce feasible solution.

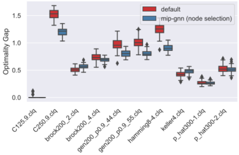
Experimental results

- CPLEX used as a baseline (version 12.10.0)
- two problem datasets :
  - Generalized independent set problem (GISP) (10 problem sets, 1000 training instances, 100 testing)
  - Fixed-charge multi-commodity network flow problem (FCMNF) (1 problem set, 1000 training instances, 100 testing)
- Feature dimension of 64
- 4 interleaved v-to-c and c-to-v passes followed by a 4-layer MLP

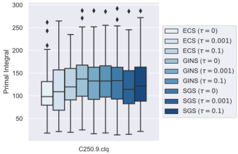
Experimental results



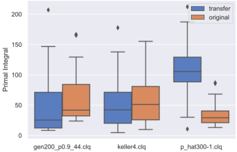
(a) Box plots for the distribution of **Primal Integrals** for the ten problem sets, each with 100 instances; lower is better.



(b) Box plots for the distribution of the **Optimality Gaps** at termination for all problem sets; lower is better.



(c) Comparison of three GNN architectures with three values for the threshold τ used during training on a single problem set from GISP; lower primal integral values are better. The performance impact of the threshold depends on the GNN architecture, with a more pronounced effect for the EdgeConvolution architecture (ECS).



(d) Transfer learning performance, GISP; Box plots for the distribution of **Primal Integrals** for three of the problem sets; lower is better. “original” refers to the performance of a model trained on instances from the same distribution, whereas “transfer” refers to that of a model trained on another distribution.

Conclusions

- Node selection :
  - GISP : improved *primal integral*, improved quality of the best solution
  - FCMNF : better solution in 81% of test instances, smaller primal integral in 62%
- Warm starting
  - better final solution in 6/9 GISP datasets
  - better optimality gap
  - **basic** warm starting