A GNN-GUIDED PREDICT-AND-SEARCH FRAMEWORK FOR MIXED-INTEGER LINEAR PROGRAMMING SUPPLEMENTARY MATERIALS

Anonymous authors
Paper under double-blind review

I EVALUATION CONFIGURATIONS

All evaluations are performed under the same configuration. The evaluation machine has two Intel(R) Xeon(R) Gold 5117 CPUs @ 2.00GHz, 256GB ram and two Nvidia V100 GPUs. SCIP 8.0.1 Bestuzheva et al. (2021), Gurobi 9.5.2 Gurobi Optimization, LLC (2022) and PyTorch 1.10.2 Paszke et al. (2019) are utilized in our experiments. The emphasis for Gurobi and SCIP is set to focus on finding better primal solutions. The time limit for running each experiment is set to $1,000$ seconds due to a tail-off of solution qualities after this point of time.

II DATA COLLECTION

The training process requires bipartite graph representations of MILP problems as the input and weighted conditional marginal probabilities of variables as the label. We first extract the bipartite graph by embedding variables and constraints as respective feature vectors (see v). Then, we run a single-thread Gurobi with a time limit of 3,600 seconds on training sets to collect feasible solutions along with their objective values. These solutions are thereafter weighted by the energy function as in Equation (**??**) to obtain weighted conditional marginal probabilities.

III COMPARING OBJECTIVE VALUES IN DIFFERENT PARTIAL SOLUTIONS

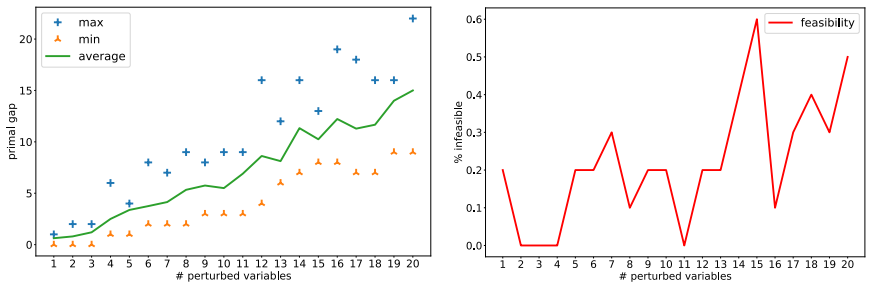
We also scrutinize why fixing strategy could fail. Variables in the optimal solution are randomly perturbed to simulate a real-world setting that a prediction is very likely to be inaccurate. Figure 1 exhibited a trend that, as we perturb more variables, the objective value gap (black) increases, and the percentage of infeasible sub-problems (red) increases. The absolute gaps shown in Figure 1a indicate large performance drawbacks given that the optimal objective value is 685. Convincingly, we conclude that fixing approaches presumably produce sub-optimal solutions. Besides, as shown in Figure 1b, randomly perturbing one variable can result in 20% of infeasible sub-problems. That is, fixing strategy could lead to infeasible sub-problems even if relatively accurate predictions are provided.

IV COMPARING WITH NEURAL DIVING PLUS SELECTIVE NET (NAIR ET AL., 2020)

Yet another interesting baseline could be the direct comparison between our proposed framework and the Neural Diving framework with Selective Net (Nair et al., 2020). However, since detailed settings and codes of the original work are not publicly available, reproducing the exact same result is almost impossible. To our best effort, a training protocol with fine parameter tuning and an evaluation process are established following algorithms provided in Nair et al. (2020).

Among six tested benchmark datasets, only three of them are publicly available: *Corlat* (Gomes et al., 2008; Conrad et al., 2007), *Neural Network Verification* (Cheng et al., 2017; Tjeng et al., 2017), and *MipLib* (Gleixner et al., 2021). Most Corlat instances can be solved by SCIP within a few seconds; MipLib contains instances with integer variables rather than binary variables, which is out of the scope of this work. Hence, Neural Network Verification (NNV) is chosen as the benchmark dataset to evaluate the implemented Selective Net aided Neural Diving framework.

It is noteworthy that, empirically, turning on the presolve option in SCIP (Bestuzheva et al., 2021) causes false assertion of feasibility on many NNV instances. Hence, in our experiments on the NNV



(a) WA (b) WA

Figure 1: This plot shows how qualities of solutions to sub-problems vary as variables are randomly perturbed in one WA instance. Maximum, minimum and average values are presented in the plot. The x-axis is the number of variables perturbed in the partial solution while y-axis of Figure 1a is the absolute gap between average objective values (only include feasible sub-problems) and the optimal objective value; y-axis of Figure 1b on the right is the percentage of infeasible sub-problems.

dataset, the presolve option is turned off, which potentially hurts the performances of both SCIP itself and frameworks implemented with SCIP. Under such circumstances, the best performance obtained is exhibited in Figure (2). Clearly, the implemented Neural Diving framework achieves significant advantages over SCIP.

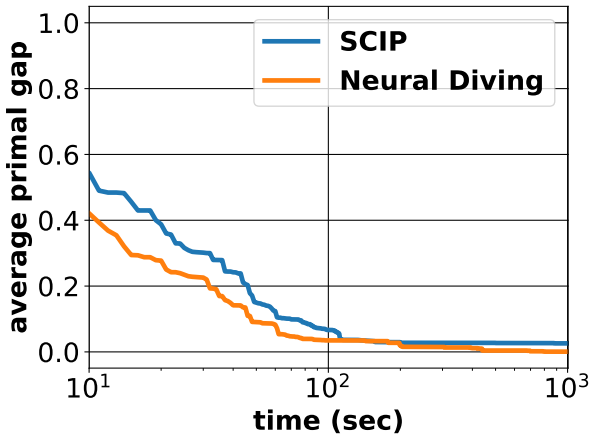
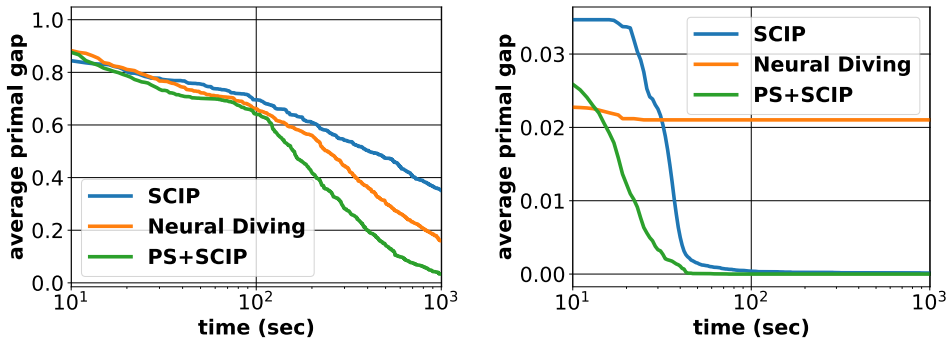


Figure 2: This figure shows the performance comparision between SCIP and our implemented Neural Diving framework. Advantages of the implemented framework can be easily observed in the aspect of both solving efficiencies and solution qualities.

With such an implementation, we can start to compare our proposed framework against the Neural Diving approach. Due to the limitation of computational power, we are not able to find suitable settings of parameters to train Neural Diving framework on WA and CA datasets. Hence we conducted experiments only on IP and IS datasets. As shown in Figure (3), the predict-and-search framework achieved at least three times better average primal gap. An interesting observation is that Neural Diving framework failed to surpass SCIP on IS dataset where the optimality is hard to achieve, while our framework outperformed both SCIP and the implemented Neural Diving method.

V FEATURE DESCRIPTIONS FOR VARIABLE NODES, CONSTRAINT NODES AND EDGES

To encode the information of MILP problems, we propose a set of features extracted from constraints, variables, and edges. This set of features is relatively light-weighted and generalized; each feature is obtained either directly from the original MILP model or by conducting simple calcula-



(a) IP (b) IS

Figure 3: Performance comparisons between PS and Neural Diving framework, where both approaches are implemented with SCIP. The average primal gap is averaged across 100 instances; each plot represents one benchmark dataset. The result shows solid performance advantages of our proposed method over Neural Diving framework.

tions. Moreover, such extractions do not require pre-solving the MILP instance, which would save a significant amount of time for large and difficult MILP problems.

Table 1: features in embedded bipartite representations

	# features.	name	detail
Variable	1	obj	normalized coefficient of variables in the obj. function
	1	v_coeff	average coefficient of the variable in all constraints
	1	Nv_coeff	degree of variable node in the bipartite representation
	1	max_coeff	maximum value among all coefficients of the variable
	1	min_coeff	minimum value among all coefficients of the variable
	1	int	binary representation to show if the variable is an integer variable
	12	pos_emb	binary encoding of the order of appearance for each variable among all variables.
Constraint	1	c_coeff	average of all coefficients in the constraint
	1	Nc_coeff	degree of constraint nodes in the bipartite representation
	1	rhs	right hand side value of the constraint
	1	sense	the sense of the constraint
Edge	1	coeff	Coefficient of variables in constraints

VI INSTANCE SIZES OF BENCHMARK PROBLEMS

Table 2 exhibits dimensions of the largest instance for each tested benchmark dataset. The numbers of constraints, variables, and binaries are presented.

Table 2: max problem sizes of each dataset

dataset	# constr.	# var.	# binary var.
IP	195	1,083	1,050
WA	64,480	61,000	1,000
IS	600	1,500	1,500
CA	6,396	1,500	1,500

VII PARAMETRIC SETTINGS FOR EXPERIMENTS

For experiments where our predict-and-search (PS) framework is compared with SCIP, Gurobi, and fixing-based strategy, the settings for fixing-parameter (k_0, k_1) and the neighborhood parameter Δ are listed in Table 3. Based on the performance of the under-laying solver, various settings of (k_0, k_1) are used to carry out experiments for each benchmark dataset shown in Table 3. The radius of the search area Δ is chosen respectively to different implementations (PS+SCIP, PS+Gurobi, and Fix+SCIP) of our framework as shown in Table 3.

Table 3: k_0, k_1, and Δ settings for different dataset

dataset	PS+SCIP		PS+Gurobi		Fixing+SCIP	
	k_0, k_1	Δ	k_0, k_1	Δ	k_0, k_1	Δ
IP	400, 5	1	400, 5	10	400, 5	0
WA	0, 500	5	0, 500	10	0, 500	0
IS	300, 300	15	300, 300	20	300, 300	0
CA	400, 0	10	600, 0	1	600, 0	0

REFERENCES

Ksenia Bestuzheva, Mathieu Besançon, Wei-Kun Chen, Antonia Chmiela, Tim Donkiewicz, Jasper van Doornmalen, Leon Eifler, Oliver Gaul, Gerald Gamrath, Ambros Gleixner, Leona Gottwald, Christoph Graczyk, Katrin Halbig, Alexander Hoen, Christopher Hojny, Rolf van der Hulst, Thorsten Koch, Marco Lübbecke, Stephen J. Maher, Frederic Matter, Erik Mühmer, Benjamin Müller, Marc E. Pfetsch, Daniel Rehfeldt, Steffan Schlein, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Boro Sofranac, Mark Turner, Stefan Vigerske, Fabian Wegscheider, Philipp Wellner, Dieter Weninger, and Jakob Witzig. The SCIP Optimization Suite 8.0. ZIB-Report 21-41, Zuse Institute Berlin, December 2021. URL `http://nbn-resolving.de/urn:nbn:de:0297-zib-85309`.

Chih-Hong Cheng, Georg Nührenberg, and Harald Ruess. Maximum resilience of artificial neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, pp. 251–268. Springer, 2017.

Jon Conrad, Carla P Gomes, Willem-Jan van Hoeve, Ashish Sabharwal, and Jordan Suter. Connections in networks: Hardness of feasibility versus optimality. In *International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming*, pp. 16–28. Springer, 2007.

Ambros Gleixner, Gregor Hendel, Gerald Gamrath, Tobias Achterberg, Michael Bastubbe, Timo Berthold, Philipp Christophel, Kati Jarck, Thorsten Koch, Jeff Linderoth, et al. Miplib 2017: data-driven compilation of the 6th mixed-integer programming library. *Mathematical Programming Computation*, 13(3):443–490, 2021.

Carla P Gomes, Willem-Jan van Hoeve, and Ashish Sabharwal. Connections in networks: A hybrid approach. In *International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming*, pp. 303–307. Springer, 2008.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2022. URL `https://www.gurobi.com`.

Vinod Nair, Sergey Bartunov, Felix Gimeno, Ingrid von Glehn, Pawel Lichocki, Ivan Lobov, Brendan O'Donoghue, Nicolas Sonnerat, Christian Tjandraatmadja, Pengming Wang, Ravichandra Addanki, Tharindi Hapuarachchi, Thomas Keck, James Keeling, Pushmeet Kohli, Ira Ktena, Yujia Li, Oriol Vinyals, and Yori Zwols. Solving mixed integer programs using neural networks, 2020.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Vincent Tjeng, Kai Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. *arXiv preprint arXiv:1711.07356*, 2017.