

---

# Learning to Condition: A Neural Heuristic for Scalable MPE Inference

## Supplementary Material

---

Anonymous Author(s)

Affiliation

Address

email

### 1 A Description of the probabilistic graphical models

2 Table 1 summarizes the networks used in our experiments, listing the number of variables and factors  
3 in each. The experiments were conducted on high-treewidth probabilistic graphical models (PGMs)  
4 of varying sizes, with the number of variables ranging from 90 to 1440 and factor counts covering a  
5 similar range.

Table 1: Summary of networks used in our experiments, including the number of variables and factors for each.

Network	Number of Factors	Number of Variables
BN 12	90	90
BN 9	105	105
BN 13	125	125
Grid 20	1200	400
BN 65	440	440
BN 59	540	540
BN 53	561	561
BN 49	661	661
BN 61	667	667
BN 45	880	880
Promedas 68	1022	1022
Promedas 60	1076	1076
BN 30	1156	1156
BN 32	1440	1440

### 6 B Hyperparameter details

7 We used a consistent set of hyperparameters across all networks in our experiments. From the initial  
8 dataset of 14,000 samples, we allocated 13,000 for training and 1,000 for testing. The training set  
9 was further split into 12,000 samples for training and 1,000 for validation.

10 The neural networks in L2C-OPT and L2C-RANK use 256-dimensional embeddings, two multi-head  
11 attention layers, and 15 skip-connection blocks. Each dense layer contains 512 units and applies a  
12 dropout rate of 0.1. The neural networks were trained using the Adam optimizer with a learning rate  
13 of  $8 \times 10^{-4}$  and an exponential decay factor of 0.97. Training was performed for up to 50 epochs  
14 with a batch size of 128, employing early stopping if validation performance did not improve for 5  
15 consecutive epochs. After training, we evaluated the model on 1,000 MPE queries with a query ratio  
16 of 0.75, defined as the fraction of variables in the query set.

Hyperparameters were selected via 5-fold cross-validation. In particular, the weighting parameters  $\lambda_{opt}$  and  $\lambda_{rank}$  were critical for balancing the preservation of optimal solutions with inference efficiency. We searched over  $\lambda_{opt} \in \{0.3, 0.35, 0.4, 0.45, 0.5\}$  and set  $\lambda_{rank} = 1 - \lambda_{opt}$ .

## C Comparing conditioning methods by per-decision time

Table 2 reports the average time required to make a single conditioning decision across all networks. The decision times for L2C-OPT and L2C-RANK remain stable across network sizes, demonstrating their scalability. In contrast, the graph-based heuristic shows increasing latency with network size, while strong branching is significantly slower and often impractical. We imposed a 30-second timeout per decision, under which strong branching failed to return results on the largest networks, BN 30 and BN 32.

Table 2: Average decision time (in seconds). Dashed cells indicate instances where the 30-second timeout was reached.

Methods Network Name	L2C-OPT	L2C-RANK	Graph	Strong Branching
<b>BN 12</b>	0.001 $\pm$ 0.000	0.002 $\pm$ 0.001	0.001 $\pm$ 0.001	2.215 $\pm$ 0.161
<b>BN 9</b>	0.003 $\pm$ 0.002	0.006 $\pm$ 0.004	0.007 $\pm$ 0.003	1.256 $\pm$ 0.115
<b>BN 13</b>	0.001 $\pm$ 0.001	0.002 $\pm$ 0.001	0.022 $\pm$ 0.020	2.787 $\pm$ 0.114
<b>Grid 20</b>	0.006 $\pm$ 0.004	0.011 $\pm$ 0.004	0.107 $\pm$ 0.062	7.325 $\pm$ 0.117
<b>BN 65</b>	0.005 $\pm$ 0.005	0.008 $\pm$ 0.005	0.086 $\pm$ 0.053	5.871 $\pm$ 0.320
<b>BN 59</b>	0.005 $\pm$ 0.004	0.009 $\pm$ 0.005	0.097 $\pm$ 0.057	10.993 $\pm$ 1.683
<b>BN 49</b>	0.007 $\pm$ 0.005	0.011 $\pm$ 0.004	0.100 $\pm$ 0.059	19.713 $\pm$ 3.417
<b>BN 53</b>	0.005 $\pm$ 0.004	0.009 $\pm$ 0.005	0.095 $\pm$ 0.056	9.106 $\pm$ 0.993
<b>BN 61</b>	0.009 $\pm$ 0.004	0.013 $\pm$ 0.004	0.103 $\pm$ 0.063	20.896 $\pm$ 3.038
<b>BN 45</b>	0.011 $\pm$ 0.005	0.014 $\pm$ 0.003	0.088 $\pm$ 0.050	8.675 $\pm$ 0.733
<b>Promedas 68</b>	0.013 $\pm$ 0.004	0.016 $\pm$ 0.003	0.085 $\pm$ 0.051	4.279 $\pm$ 0.118
<b>Promedas 60</b>	0.015 $\pm$ 0.002	0.015 $\pm$ 0.002	0.089 $\pm$ 0.052	5.432 $\pm$ 0.270
<b>BN 30</b>	0.018 $\pm$ 0.002	0.018 $\pm$ 0.002	0.098 $\pm$ 0.058	—
<b>BN 32</b>	0.023 $\pm$ 0.005	0.025 $\pm$ 0.009	0.102 $\pm$ 0.060	—

## D Greedy conditioning performance using SCIP as oracle

In this section, we analyze the conditioning performance of all methods under varying oracle time budgets and numbers of greedy conditioning decisions when SCIP solver [1] is used as the oracle. For each method and network, we report the average percentage gap in log-likelihood. The y-axis in Figures 1 to 14 shows the average percentage gap in log-likelihood scores, computed as:

$$\text{avg. \% gap} = \frac{1}{N} \sum_{i=1}^N \frac{(\mathcal{LL}_S^{(i)} - \mathcal{LL}_D^{(i)})}{|\mathcal{LL}_S^{(i)}|} \times 100 \quad (1)$$

where  $\mathcal{LL}_S^{(i)}$  denotes the log-likelihood score of the oracle before conditioning, and  $\mathcal{LL}_D^{(i)}$  denotes the score after conditioning for instance  $i$ . Thus, negative values indicate that conditioning improves solution quality, while positive values indicate a decline in performance. The x-axis represents the number of conditioning decisions, and each subfigure corresponds to a distinct oracle time budget.

As shown in the figures, our neural network-based strategies, L2C-OPT and L2C-RANK, consistently improve the oracle performance after conditioning, as evidenced by negative percentage gaps. In contrast, the **graph-based heuristic** and the **full strong branching heuristic** lead to improvements in only a small fraction of instances. Since these baselines do not produce optimal decisions, their performance generally worsens with increasing oracle time budgets, resulting in larger positive average gaps. This degradation becomes more pronounced as the number of conditioning decisions

42 increases. In comparison, our approaches yield increasingly negative average gaps with more  
 43 decisions, indicating consistent performance gains.

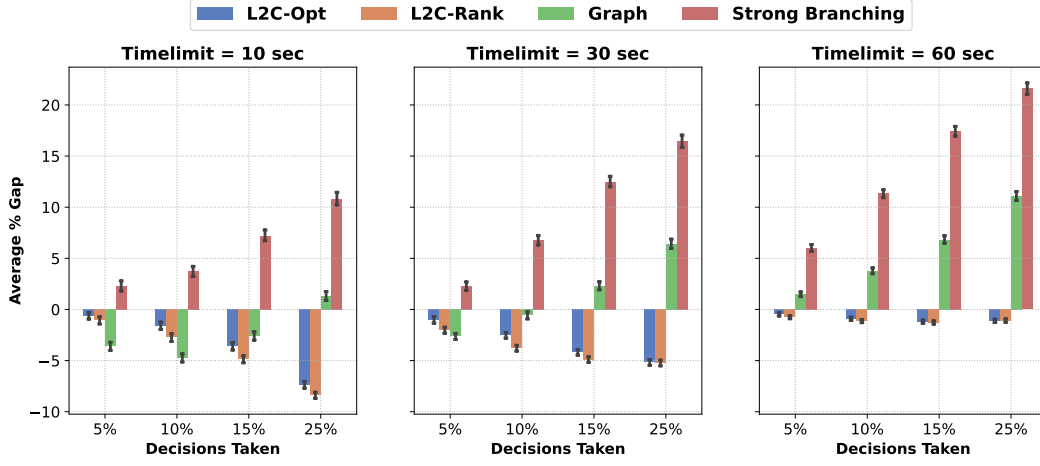


Figure 1: Average percentage gap on the BN 12 network for greedy conditioning using our methods, L2C-OPT and L2C-RANK, and baseline approaches across varying time budgets and numbers of decisions. More negative values indicate better performance.

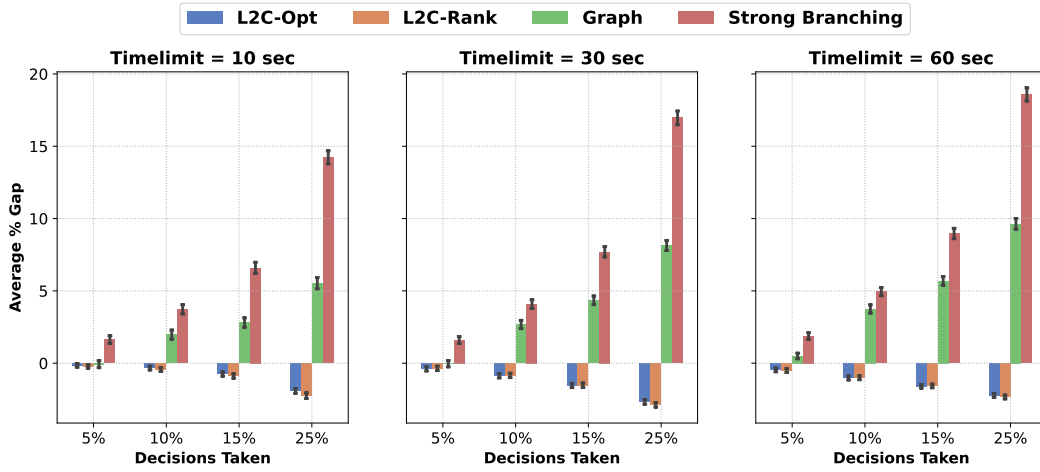


Figure 2: Average percentage gap on the BN 9 network for greedy conditioning using our methods, L2C-OPT and L2C-RANK, and baseline approaches across varying time budgets and numbers of decisions. More negative values indicate better performance.

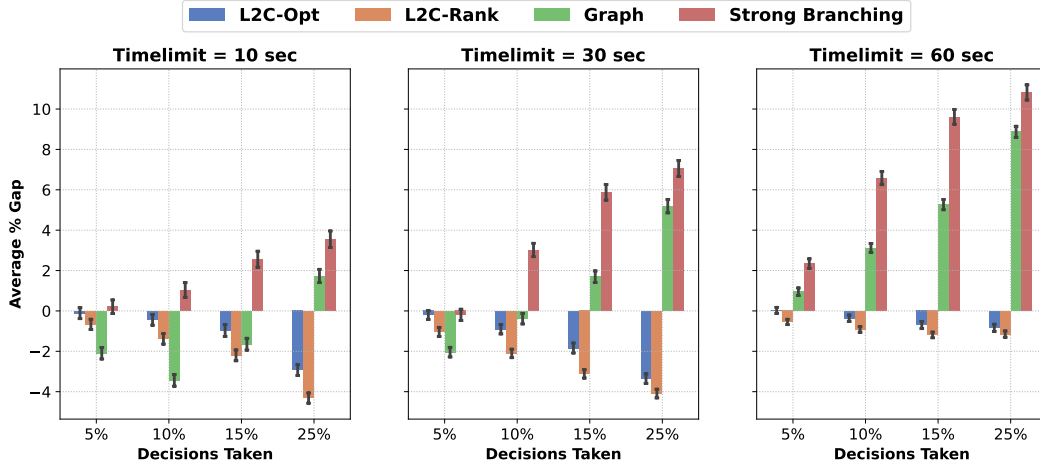


Figure 3: Average percentage gap on the BN 13 network for greedy conditioning using our methods, L2C-OPT and L2C-RANK, and baseline approaches across varying time budgets and numbers of decisions. More negative values indicate better performance.

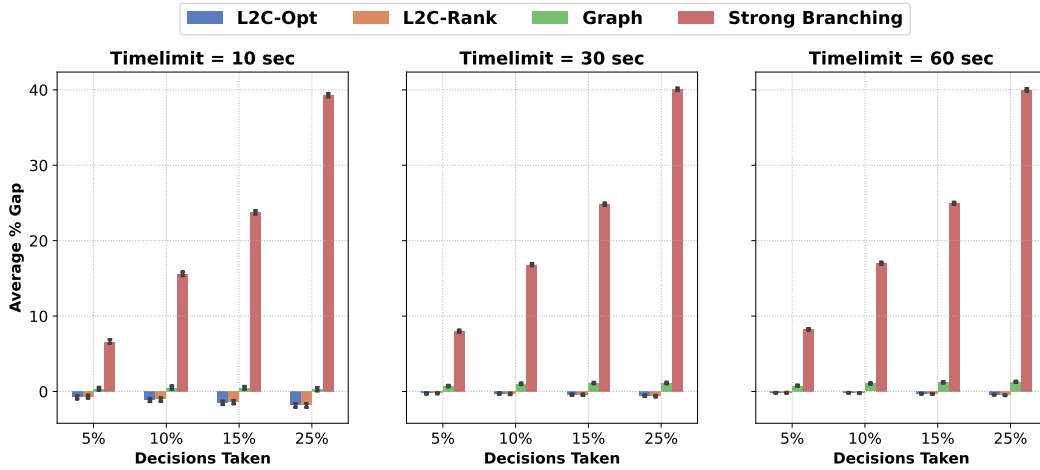


Figure 4: Average percentage gap on the Grid 20 network for greedy conditioning using our methods, L2C-OPT and L2C-RANK, and baseline approaches across varying time budgets and numbers of decisions. More negative values indicate better performance.

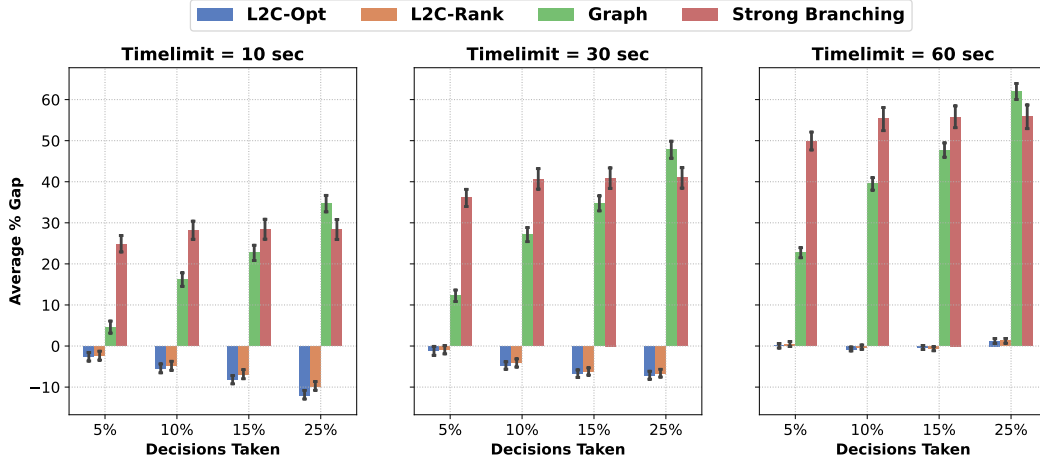


Figure 5: Average percentage gap on the BN 65 network for greedy conditioning using our methods, L2C-OPT and L2C-RANK, and baseline approaches across varying time budgets and numbers of decisions. More negative values indicate better performance.

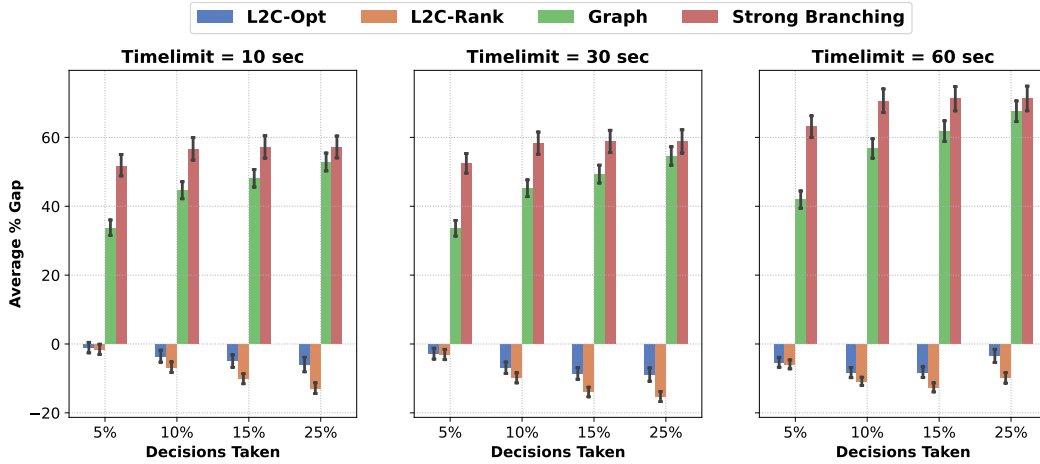


Figure 6: Average percentage gap on the BN 59 network for greedy conditioning using our methods, L2C-OPT and L2C-RANK, and baseline approaches across varying time budgets and numbers of decisions. More negative values indicate better performance.

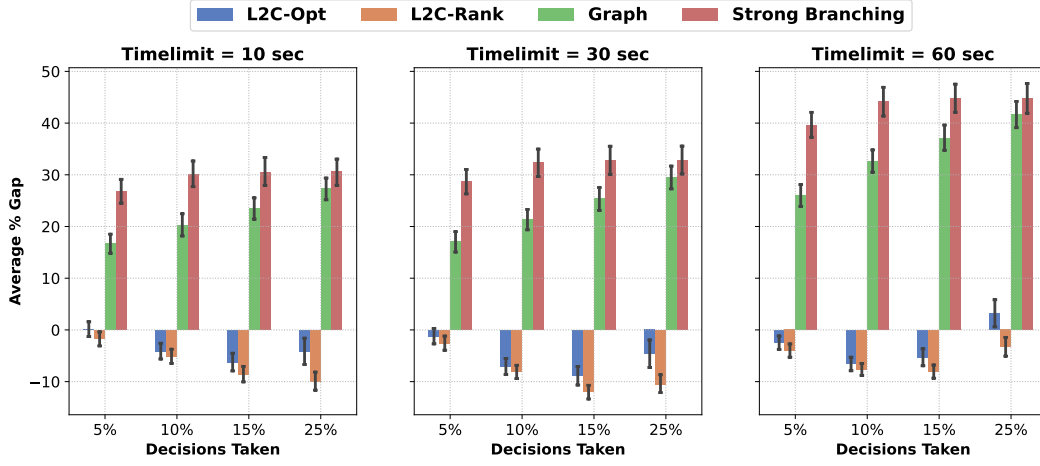


Figure 7: Average percentage gap on the BN 53 network for greedy conditioning using our methods, L2C-OPT and L2C-RANK, and baseline approaches across varying time budgets and numbers of decisions. More negative values indicate better performance.

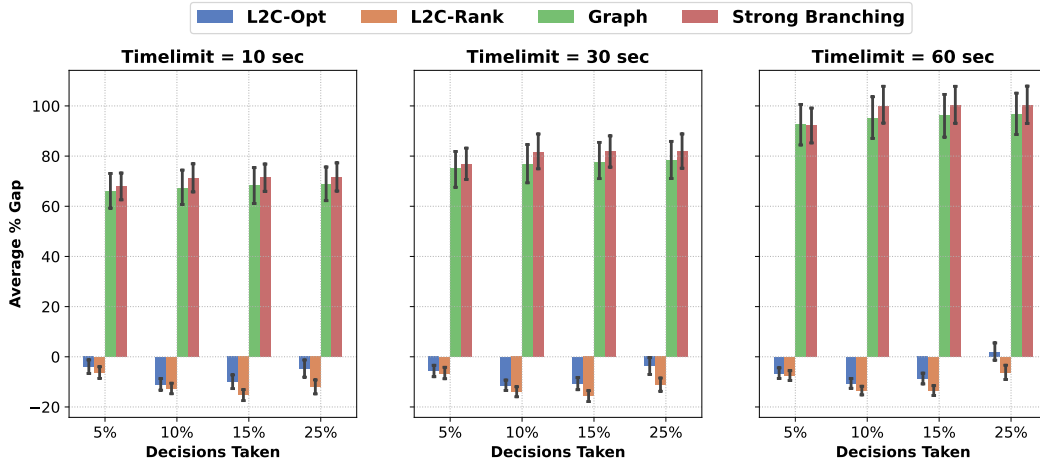


Figure 8: Average percentage gap on the BN 49 network for greedy conditioning using our methods, L2C-OPT and L2C-RANK, and baseline approaches across varying time budgets and numbers of decisions. More negative values indicate better performance.

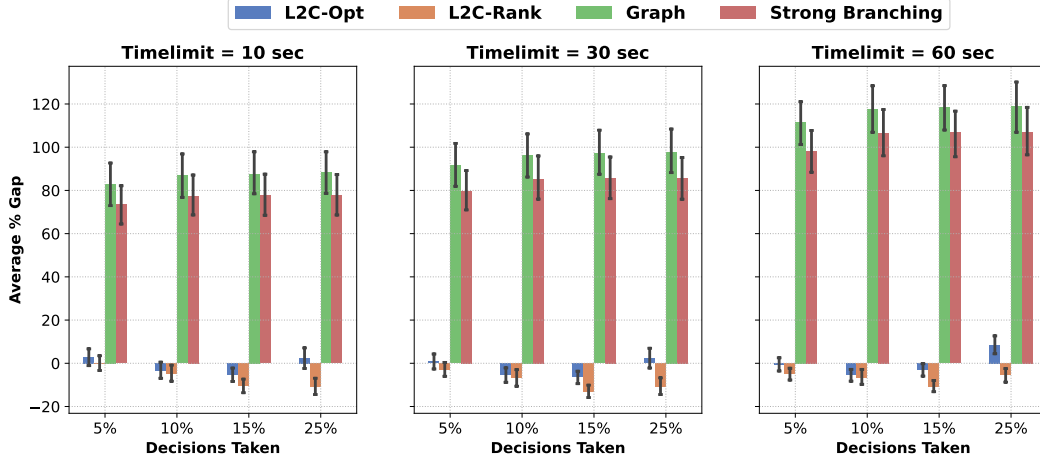


Figure 9: Average percentage gap on the BN 61 network for greedy conditioning using our methods, L2C-OPT and L2C-RANK, and baseline approaches across varying time budgets and numbers of decisions. More negative values indicate better performance.

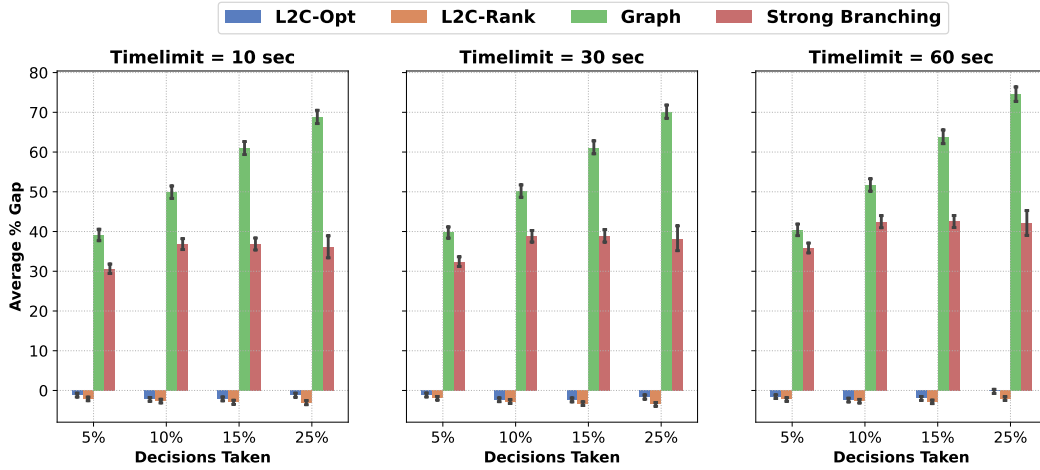


Figure 10: Average percentage gap on the BN 45 network for greedy conditioning using our methods, L2C-OPT and L2C-RANK, and baseline approaches across varying time budgets and numbers of decisions. More negative values indicate better performance.

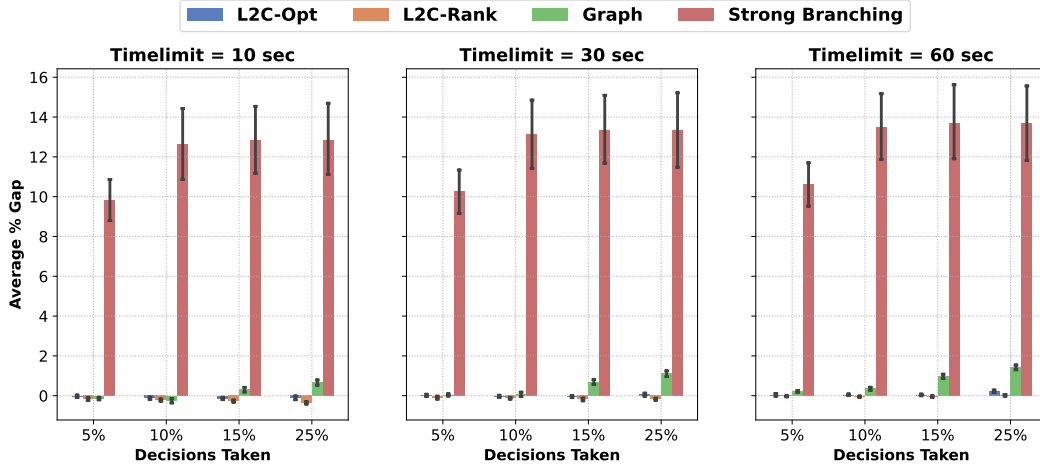


Figure 11: Average percentage gap on the Promedas 68 network for greedy conditioning using our methods, L2C-OPT and L2C-RANK, and baseline approaches across varying time budgets and numbers of decisions. More negative values indicate better performance.

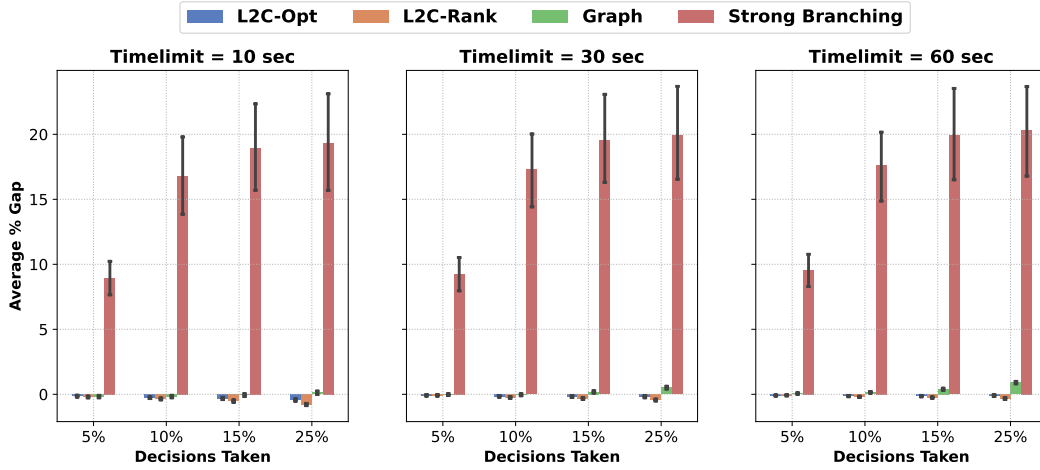


Figure 12: Average percentage gap on the Promedas 60 network for greedy conditioning using our methods, L2C-OPT and L2C-RANK, and baseline approaches across varying time budgets and numbers of decisions. More negative values indicate better performance.

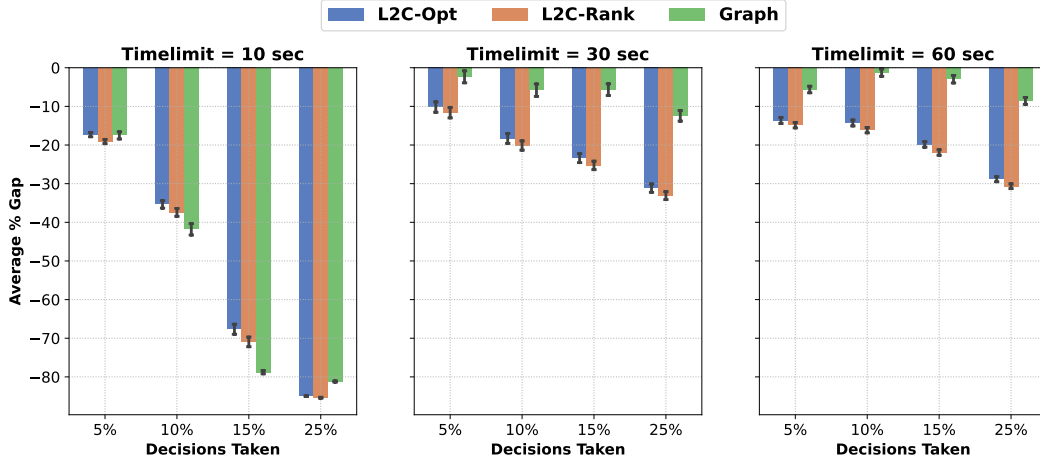


Figure 13: Average percentage gap on the BN 30 network for greedy conditioning using our methods, L2C-OPT and L2C-RANK, and baseline approaches across varying time budgets and numbers of decisions. More negative values indicate better performance.

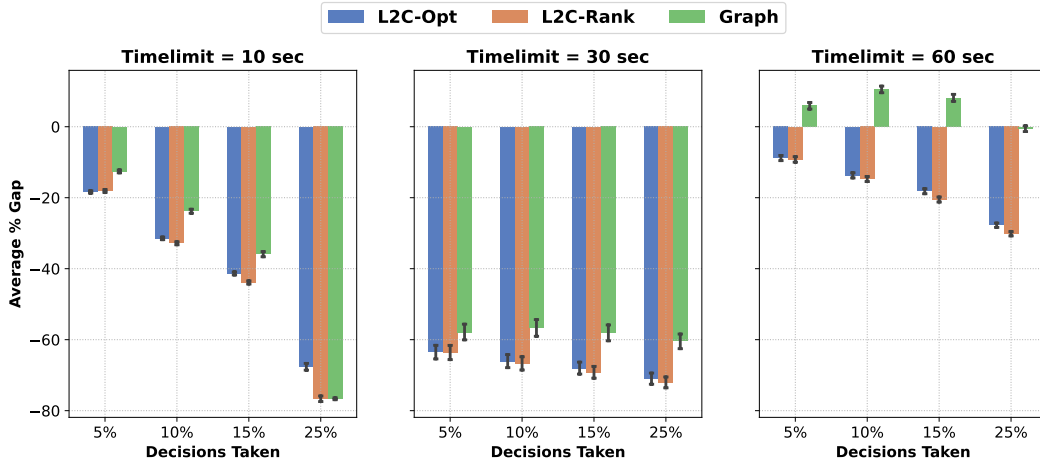


Figure 14: Average percentage gap on the BN 32 network for greedy conditioning using our methods, L2C-OPT and L2C-RANK, and baseline approaches across varying time budgets and numbers of decisions. More negative values indicate better performance.

## 44 E Conditioning performance for beam search-based conditioning

### 45 E.1 Using SCIP as oracle

46 We now present detailed results comparing the performance of various conditioning strategies  
 47 when used in beam search, using the SCIP solver as the oracle. Each bar represents the average  
 48 log-likelihood gap (computed using equation 1) before and after conditioning, aggregated over  
 49 conditioning depths of 5%, 10%, 15%, and 25% of the query variables, and averaged across all MPE  
 50 queries solved using the oracle. Each subfigure corresponds to a specific oracle time budget.

51 Our neural strategies, L2C-OPT and L2C-RANK, consistently outperform the **full strong branching**  
 52 **heuristic** in improving oracle performance, as evidenced by the negative average percentage gap. As  
 53 the beam width increases, the performance of our methods either improves or remains consistent. In  
 54 contrast, the strong branching heuristic often fails to return a decision within the 30-second time limit  
 55 on larger networks and wider beams. As a result, the corresponding bars are omitted in the plots.

56 We omit results for the **graph-based heuristic**, as it only supports a beam width of 1 and is therefore  
 57 not applicable in this setting.

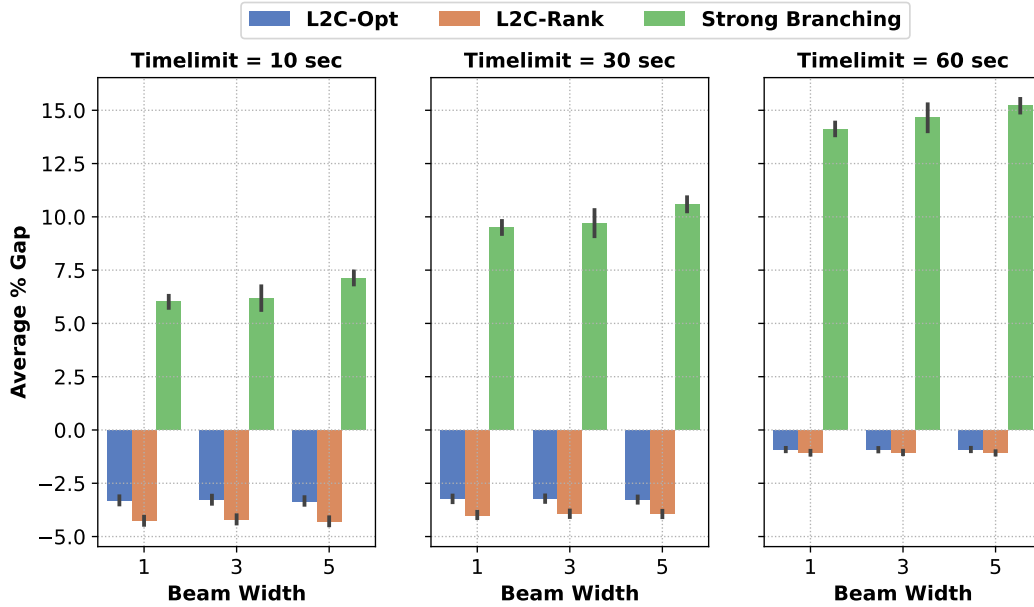


Figure 15: Average percentage gap on the BN 12 network comparing our methods to baseline approaches across varying time budgets and beam widths using beam-search based conditioning. More negative values indicate better performance.

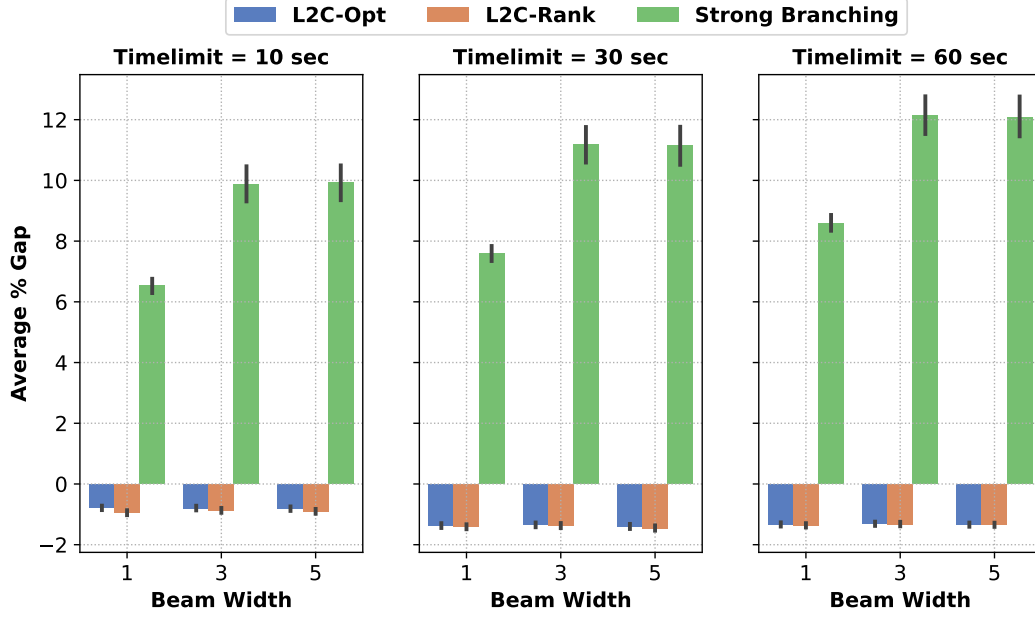


Figure 16: Average percentage gap on the BN 9 network comparing our methods to baseline approaches across varying time budgets and beam widths using beam-search based conditioning. More negative values indicate better performance.

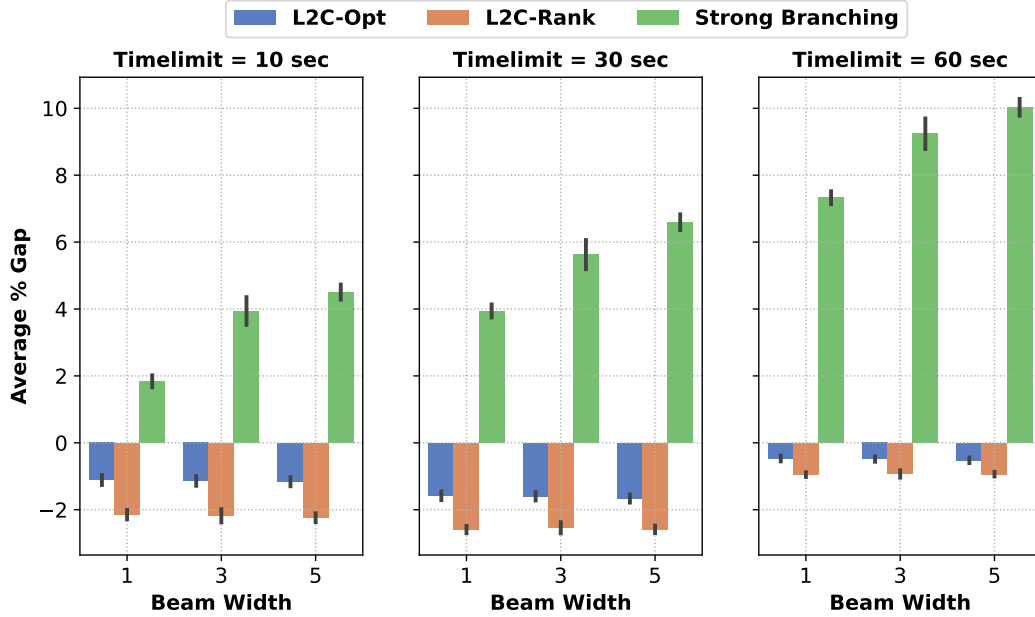


Figure 17: Average percentage gap on the BN 13 network comparing our methods to baseline approaches across varying time budgets and beam widths using beam-search based conditioning. More negative values indicate better performance.

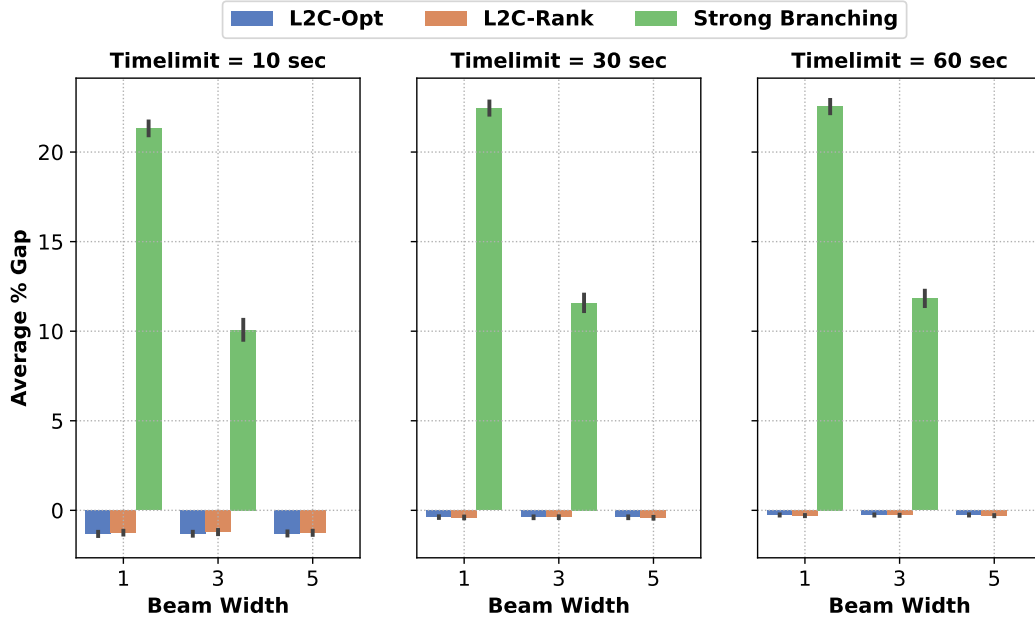


Figure 18: Average percentage gap on the Grid 20 network comparing our methods to baseline approaches across varying time budgets and beam widths using beam-search based conditioning. More negative values indicate better performance. Missing bars for strong branching indicate timeouts.

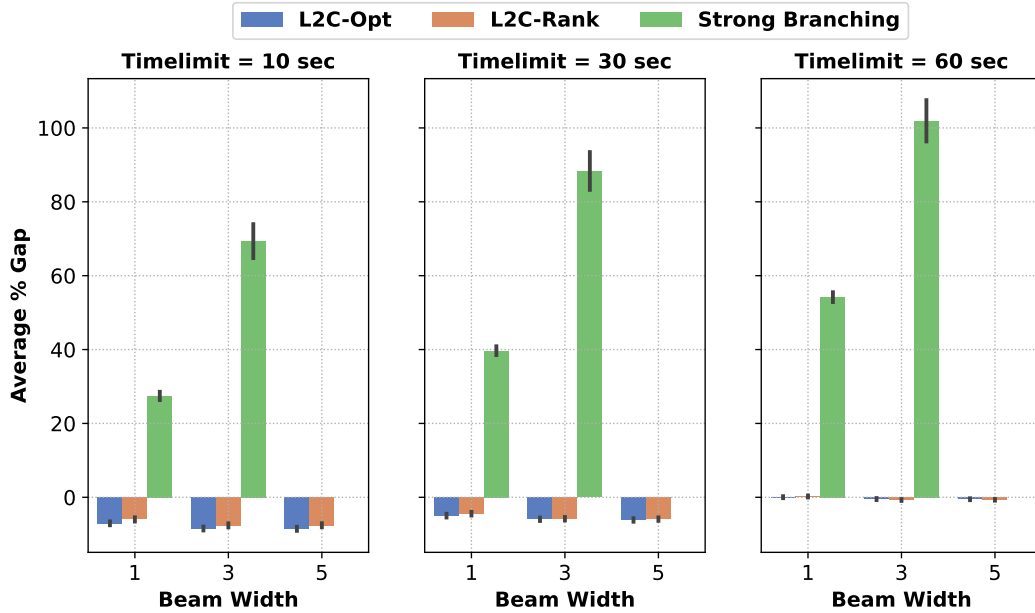


Figure 19: Average percentage gap on the BN 65 network comparing our methods to baseline approaches across varying time budgets and beam widths using beam-search based conditioning. More negative values indicate better performance. Missing bars for strong branching indicate timeouts.

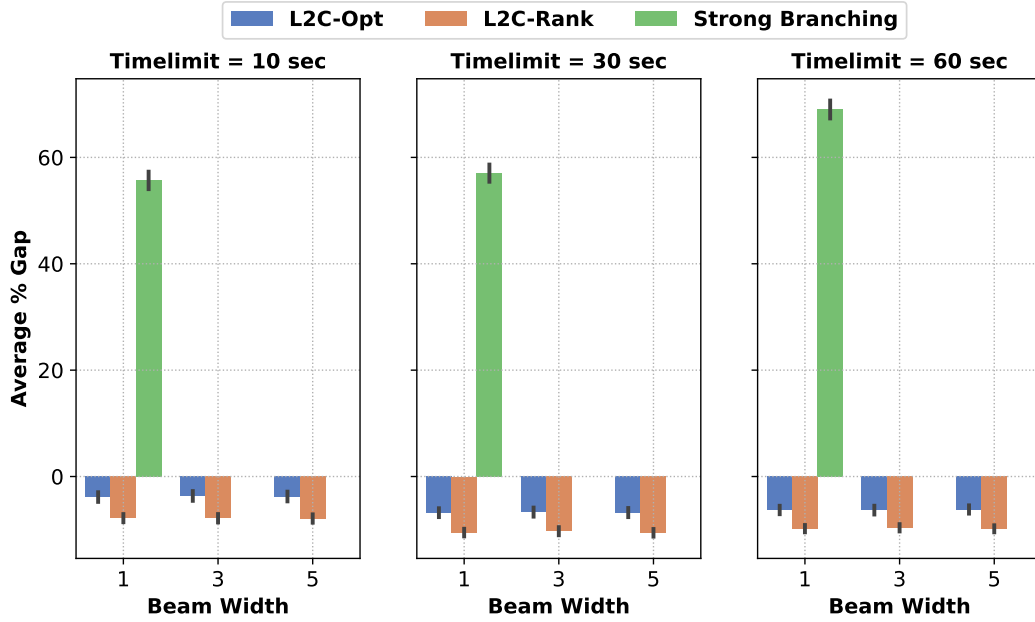


Figure 20: Average percentage gap on the BN 59 network comparing our methods to baseline approaches across varying time budgets and beam widths using beam-search based conditioning. More negative values indicate better performance. Missing bars for strong branching indicate timeouts.

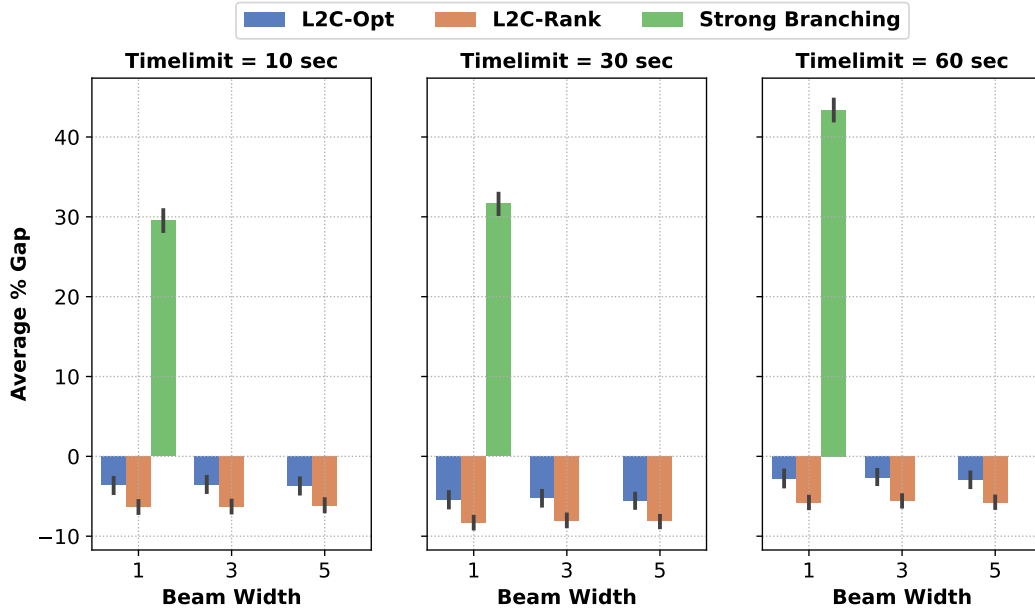


Figure 21: Average percentage gap on the BN 53 network comparing our methods to baseline approaches across varying time budgets and beam widths using beam-search based conditioning. More negative values indicate better performance. Missing bars for strong branching indicate timeouts.

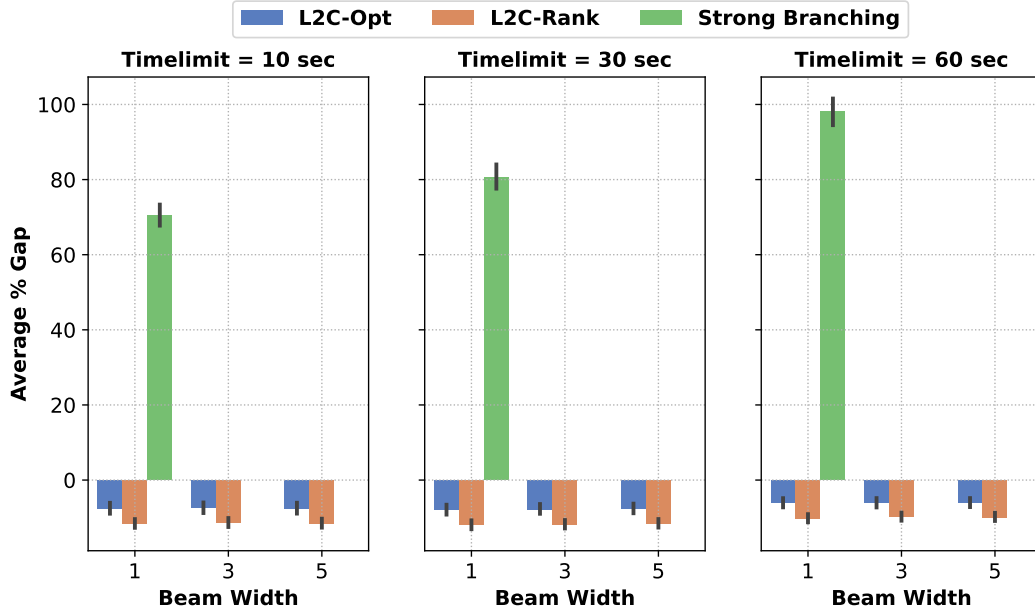


Figure 22: Average percentage gap on the BN 49 network comparing our methods to baseline approaches across varying time budgets and beam widths using beam-search based conditioning. More negative values indicate better performance. Missing bars for strong branching indicate timeouts.

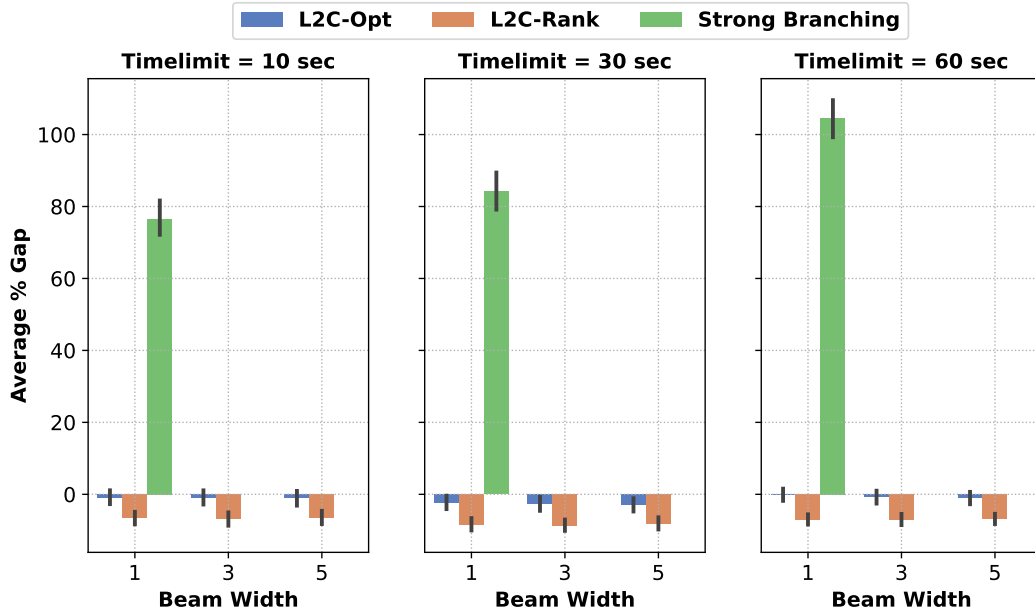


Figure 23: Average percentage gap on the BN 61 network comparing our methods to baseline approaches across varying time budgets and beam widths using beam-search based conditioning. More negative values indicate better performance. Missing bars for strong branching indicate timeouts.

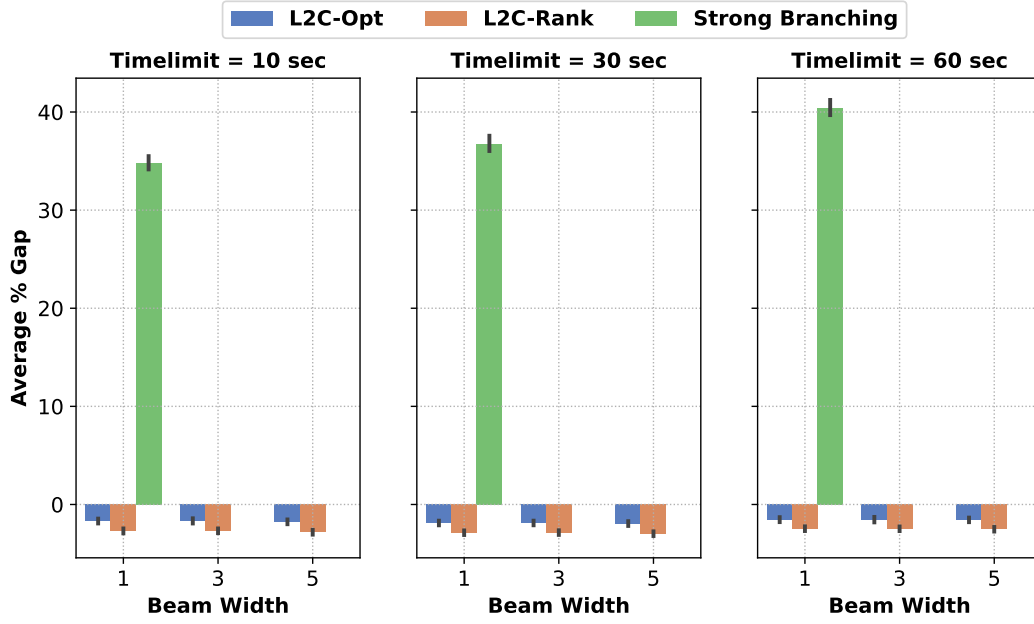


Figure 24: Average percentage gap on the BN 45 network comparing our methods to baseline approaches across varying time budgets and beam widths using beam-search based conditioning. More negative values indicate better performance. Missing bars for strong branching indicate timeouts.

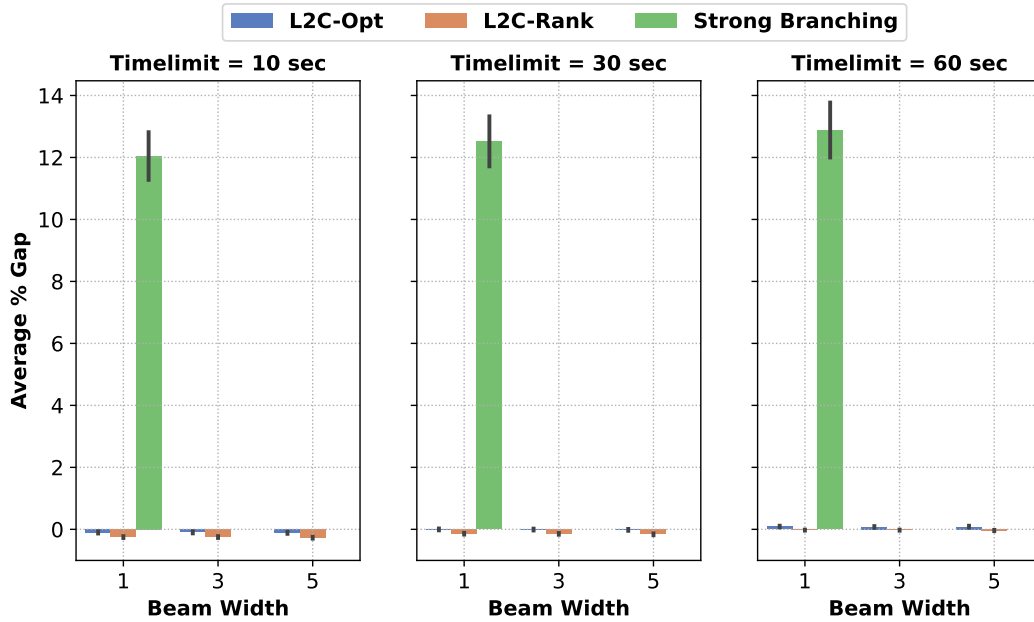


Figure 25: Average percentage gap on the Promedas 68 network comparing our methods to baseline approaches across varying time budgets and beam widths using beam-search based conditioning. More negative values indicate better performance. Missing bars for strong branching indicate timeouts.

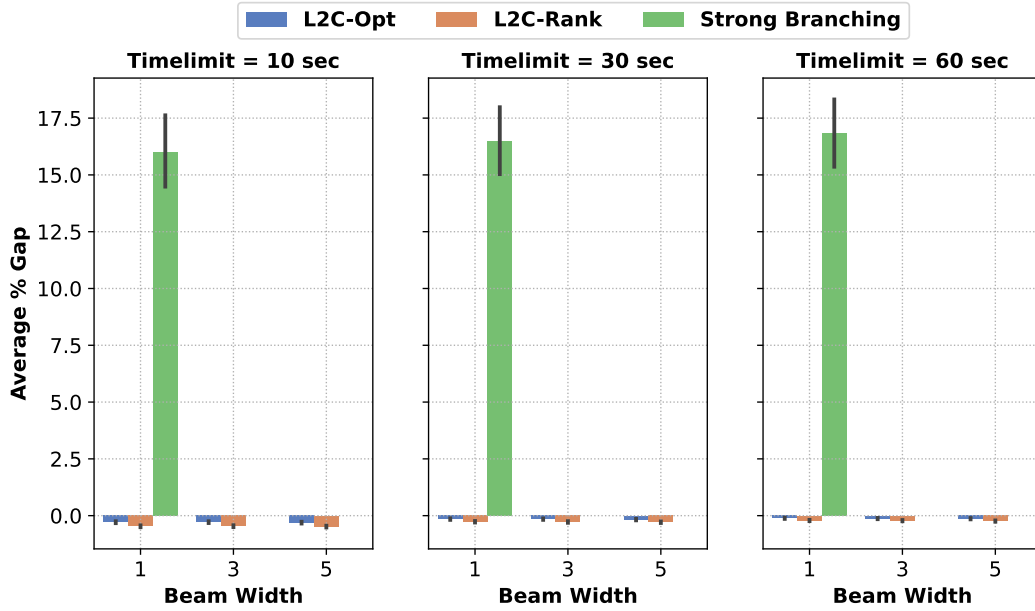


Figure 26: Average percentage gap on the Promedas 60 network comparing our methods to baseline approaches across varying time budgets and beam widths using beam-search based conditioning. More negative values indicate better performance. Missing bars for strong branching indicate timeouts.

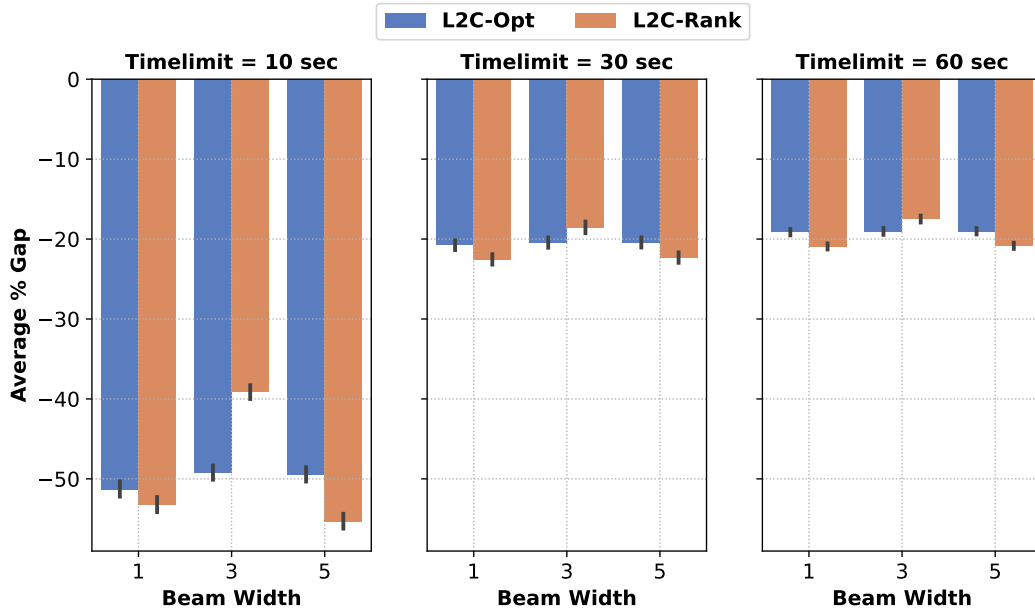


Figure 27: Average percentage gap on the BN 30 network comparing our methods to baseline approaches across varying time budgets and beam widths using beam-search based conditioning. More negative values indicate better performance. Missing bars for strong branching indicate timeouts.

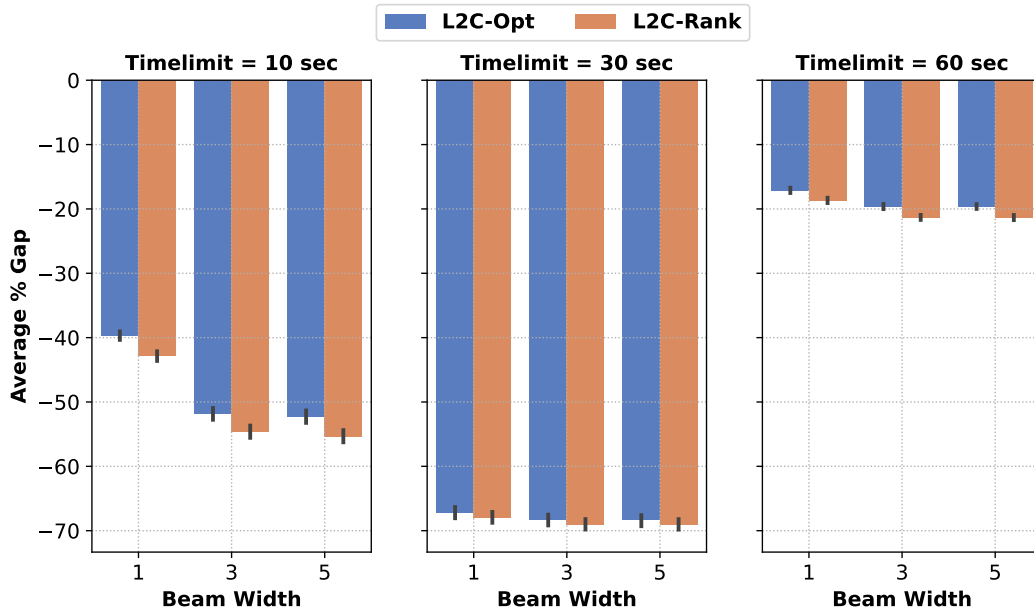


Figure 28: Average percentage gap on the BN 32 network comparing our methods to baseline approaches across varying time budgets and beam widths using beam-search based conditioning. More negative values indicate better performance. Missing bars for strong branching indicate timeouts.

## 58 E.2 Using AOBB as oracle

59 In this section, we evaluate the performance of beam search-based conditioning with AOBB (per  
60 Otten and Dechter [2] and implemented by Otten [3]) as the oracle, using beam widths of 3 and 5,  
61 as shown in Figures 29 and 30, respectively. Each figure plots the average solution gap against the  
62 percentage reduction in the number of search nodes across 1,000 MPE queries over all networks,  
63 with each point representing one network. The solution gap is computed using Equation 1, while  
64 node reduction is calculated using the same formula with log-likelihood scores replaced by node  
65 counts before and after conditioning.

66 Our methods, L2C-OPT and L2C-RANK, consistently yield lower solution gaps—indicating better  
67 preservation of optimal solutions—and higher node reductions, reflecting improved search efficiency.  
68 In contrast, the full strong branching heuristic frequently fails to preserve solution quality, as indicated  
69 by its larger gaps, and produces fewer data points due to timeouts exceeding the 30-second limit per  
70 decision.

71 As before, we omit results for the **graph-based heuristic**, as it supports only a beam width of 1 and  
72 is therefore not applicable in this setting.

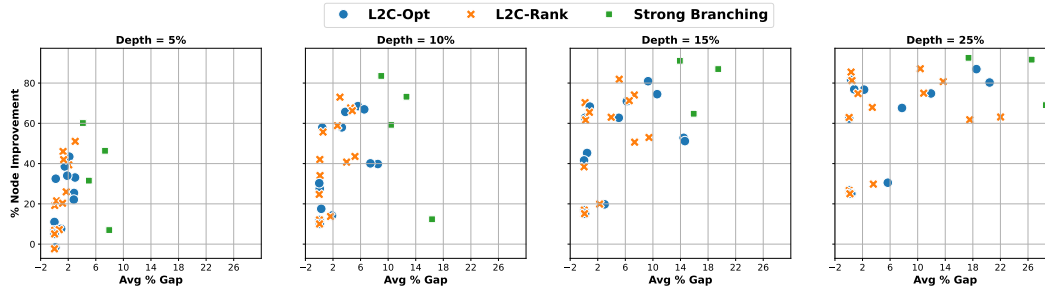


Figure 29: Beam search-based conditioning with AOBB as the oracle using a beam width of 3. Each subfigure corresponds to a fixed number of conditioning decisions. The x-axis indicates the average solution gap (lower is better), and the y-axis indicates the percentage reduction in node count (higher is better). Each point represents a single network.

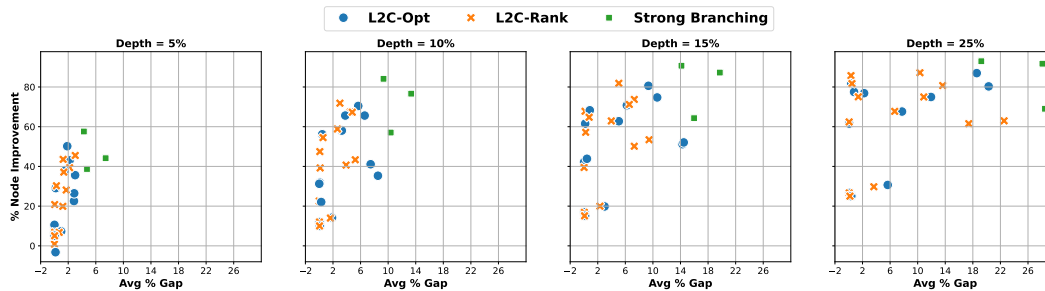


Figure 30: Beam search-based conditioning with AOBB as the oracle using a beam width of 5. Each subfigure corresponds to a fixed number of conditioning decisions. The x-axis indicates the average solution gap (lower is better), and the y-axis indicates the percentage reduction in node count (higher is better). Each point represents a single network.

## 73 F Incorporating L2C scores as branching and node selection heuristics in 74 branch-and-bound

75 In this section, we compare our trained neural networks—used as branching rules [4] and node  
76 selection heuristics [5] within the SCIP framework—with SCIP’s default heuristics [1, 6, 7]. We  
77 evaluate performance based on the average percentage gap in log-likelihood (LL) scores between our  
78 methods, L2C-OPT and L2C-RANK, and SCIP’s default strategies on the same set of MPE queries.  
79 The gap is computed as:

$$\frac{1}{N} \sum_{i=1}^N \frac{\mathcal{LL}_S^{(i)} - \mathcal{LL}_N^{(i)}}{|\mathcal{LL}_S^{(i)}|} \times 100 \quad (2)$$

where  $\mathcal{LL}_S^{(i)}$  denotes the log-likelihood score achieved by SCIP's default heuristics, and  $\mathcal{LL}_N^{(i)}$  denotes the score obtained using our L2C methods on the  $i$ -th instance. Negative values indicate that our methods perform better; positive values indicate superior performance by SCIP's default heuristics.

As shown in Figures 31 to 44, the percentage gap is typically negative, demonstrating that our methods consistently yield higher log-likelihood scores than SCIP within the same time budget. This indicates that for time-constrained settings, L2C-OPT and L2C-RANK can find higher-quality solutions more efficiently than SCIP's default branching and node selection strategies. Overall, our learned heuristics not only produce better decisions but also execute faster than the state-of-the-art methods implemented in SCIP.

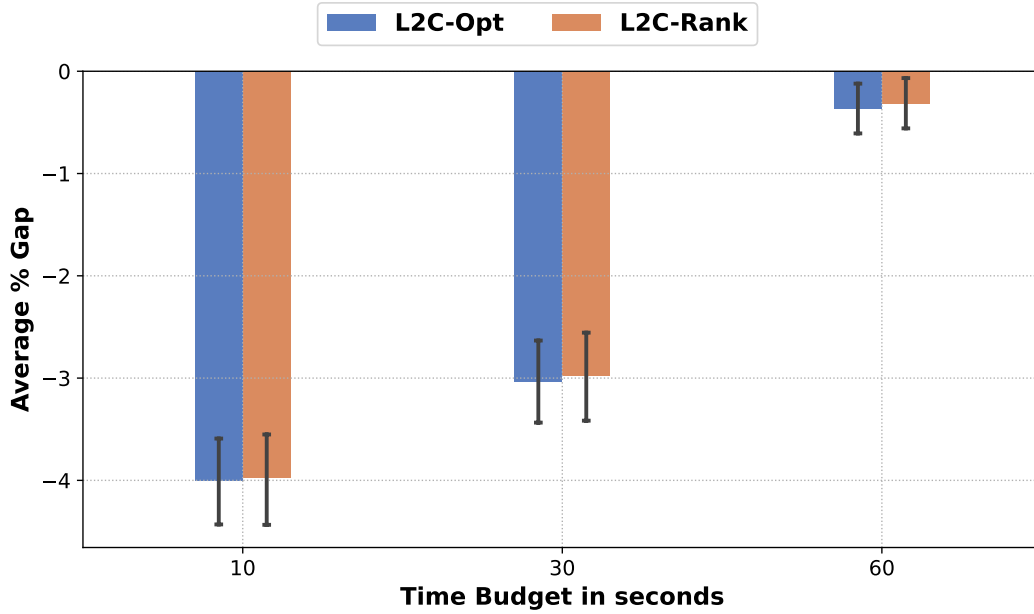


Figure 31: Comparison of SCIP's default heuristics with our neural strategies, L2C-OPT and L2C-RANK, for branching and node selection within the SCIP framework on the BN 12 network in terms of average % gap in log-likelihood. More negative values indicate better performance.

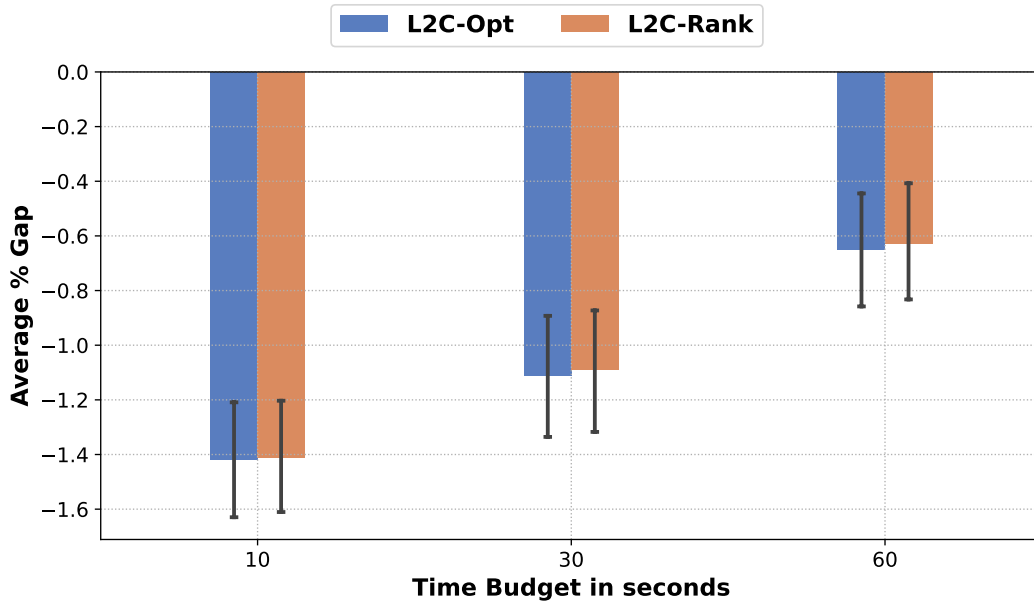


Figure 32: Comparison of SCIP’s default heuristics with our neural strategies, L2C-OPT and L2C-RANK, for branching and node selection within the SCIP framework on the BN 9 network in terms of average % gap in log-likelihood. More negative values indicate better performance.

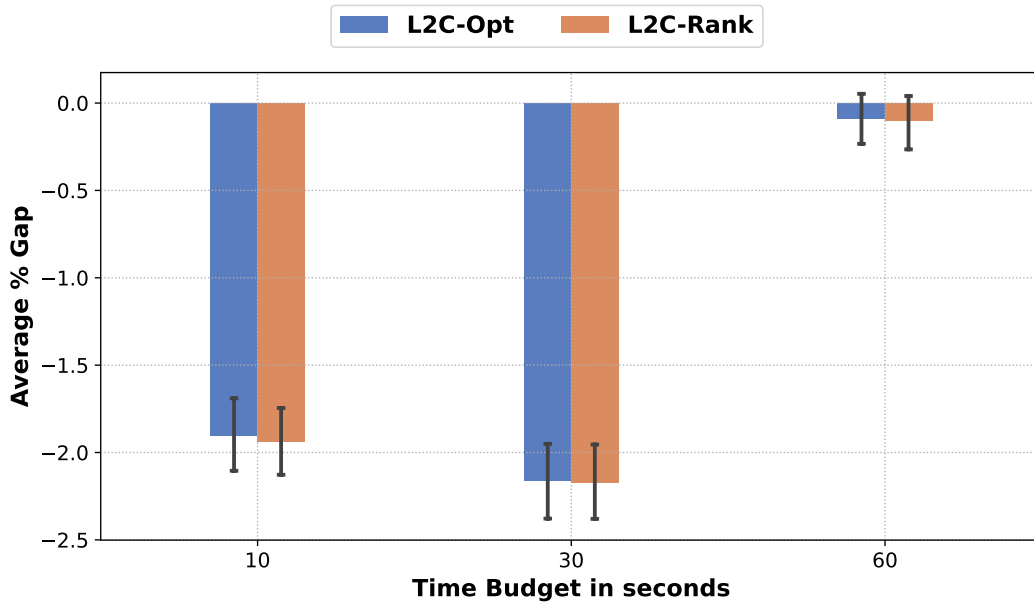


Figure 33: Comparison of SCIP’s default heuristics with our neural strategies, L2C-OPT and L2C-RANK, for branching and node selection within the SCIP framework on the BN 13 network in terms of average % gap in log-likelihood. More negative values indicate better performance.

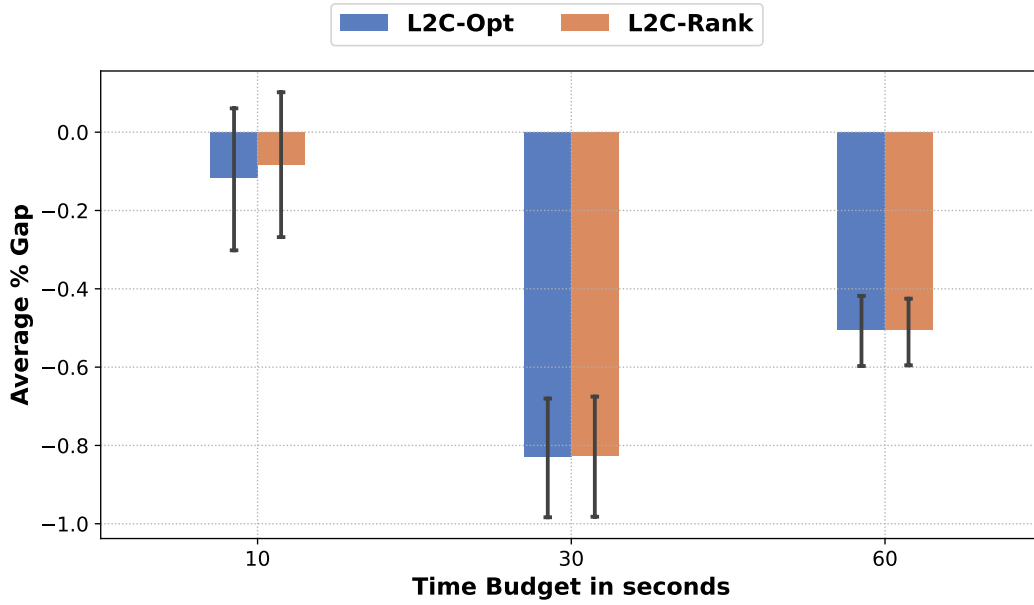


Figure 34: Comparison of SCIP’s default heuristics with our neural strategies, L2C-OPT and L2C-RANK, for branching and node selection within the SCIP framework on the Grid 20 network in terms of average % gap in log-likelihood. More negative values indicate better performance.

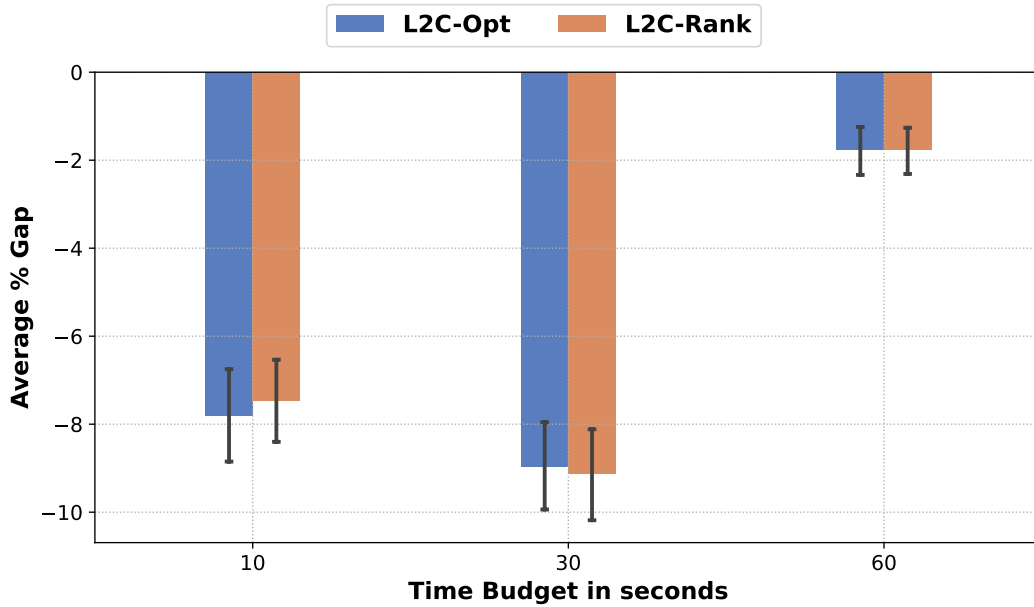


Figure 35: Comparison of SCIP’s default heuristics with our neural strategies, L2C-OPT and L2C-RANK, for branching and node selection within the SCIP framework on the BN 65 network in terms of average % gap in log-likelihood. More negative values indicate better performance.

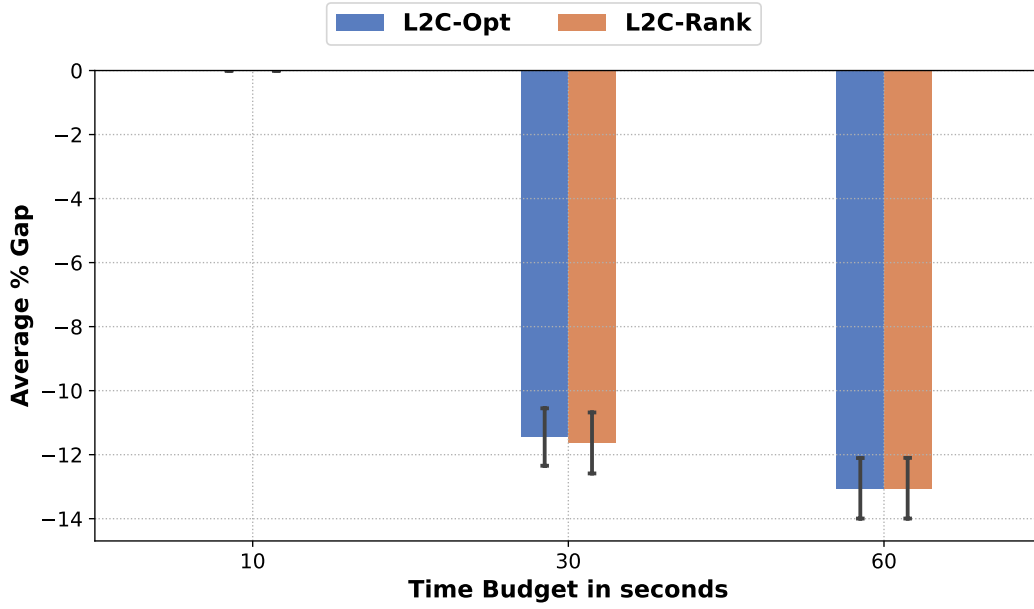


Figure 36: Comparison of SCIP’s default heuristics with our neural strategies, L2C-OPT and L2C-RANK, for branching and node selection within the SCIP framework on the BN 59 network in terms of average % gap in log-likelihood. More negative values indicate better performance.

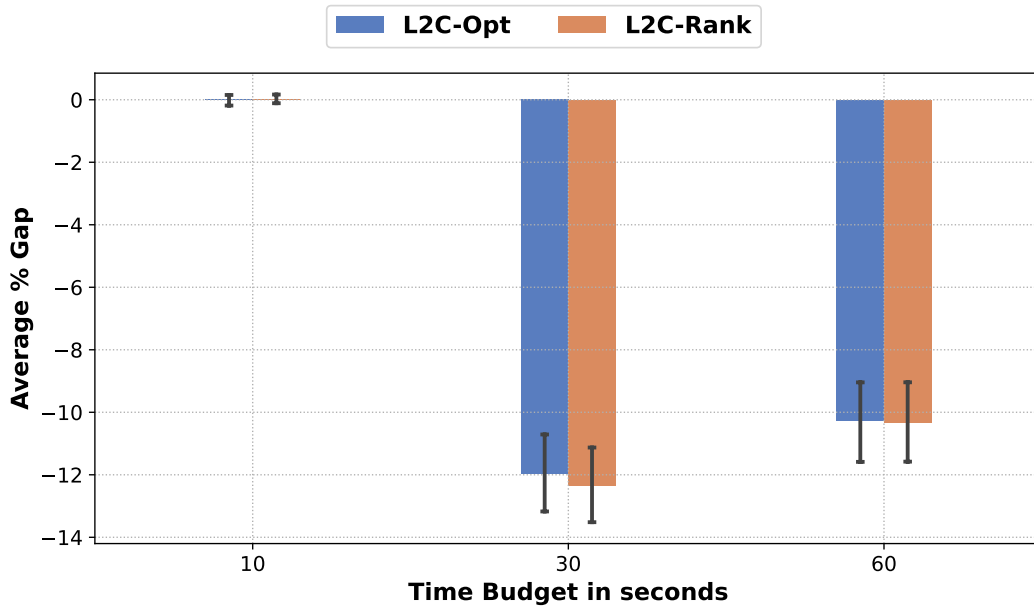


Figure 37: Comparison of SCIP’s default heuristics with our neural strategies, L2C-OPT and L2C-RANK, for branching and node selection within the SCIP framework on the BN 53 network in terms of average % gap in log-likelihood. More negative values indicate better performance.

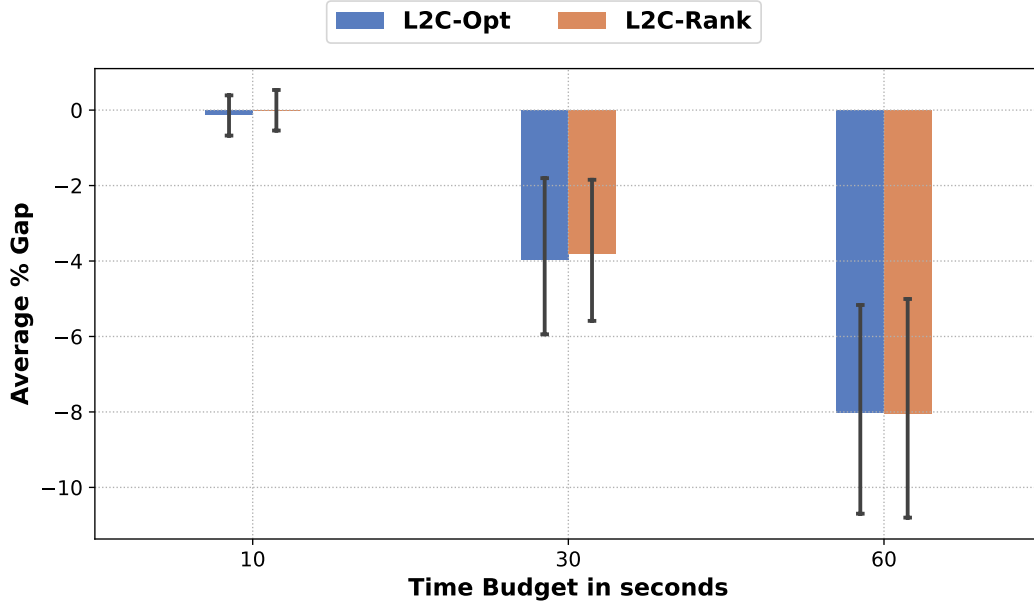


Figure 38: Comparison of SCIP’s default heuristics with our neural strategies, L2C-OPT and L2C-RANK, for branching and node selection within the SCIP framework on the BN 49 network in terms of average % gap in log-likelihood. More negative values indicate better performance.

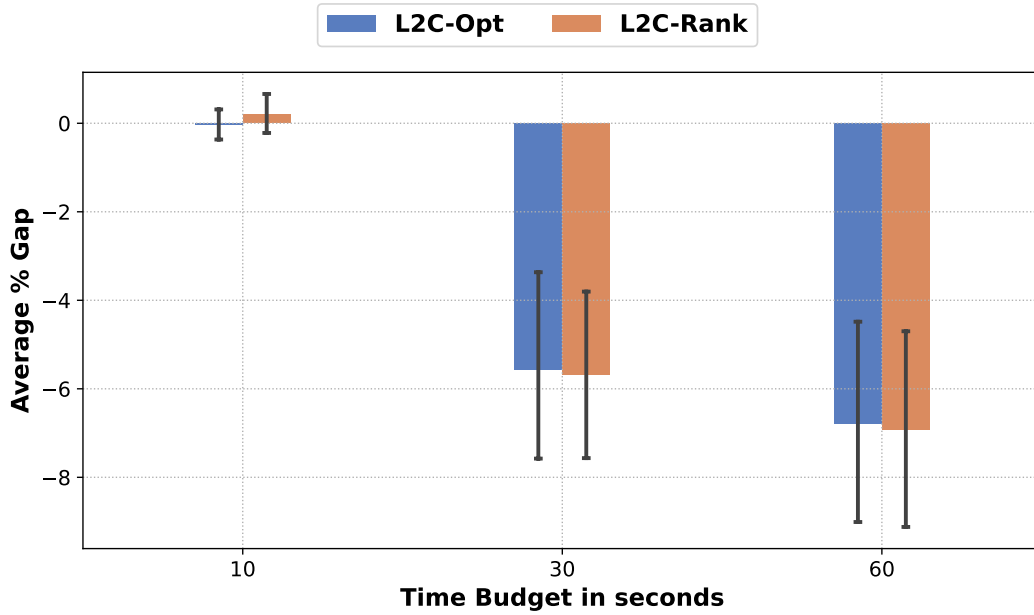


Figure 39: Comparison of SCIP’s default heuristics with our neural strategies, L2C-OPT and L2C-RANK, for branching and node selection within the SCIP framework on the BN 61 network in terms of average % gap in log-likelihood. More negative values indicate better performance.

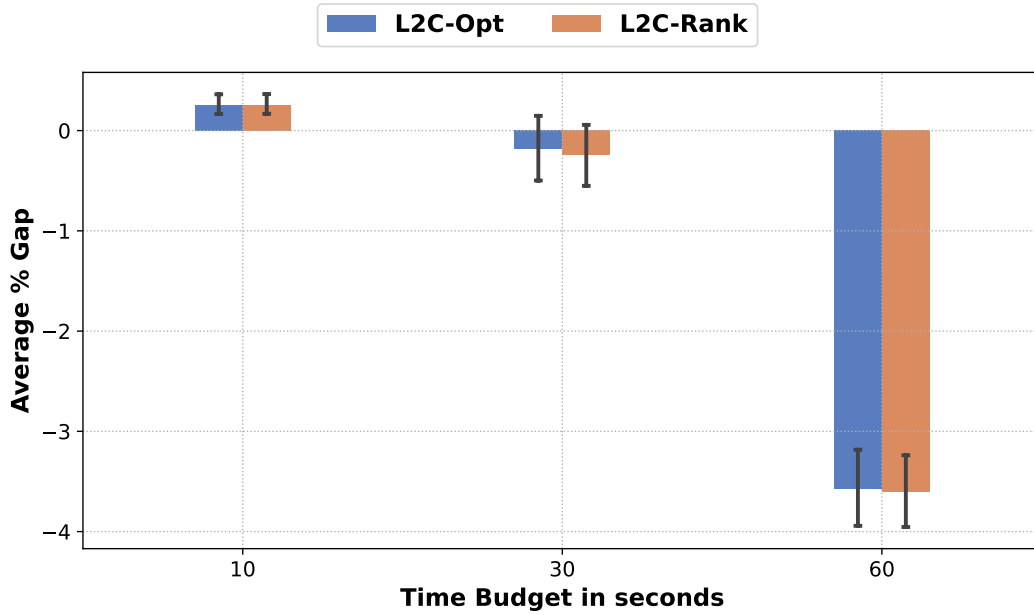


Figure 40: Comparison of SCIP’s default heuristics with our neural strategies, L2C-OPT and L2C-RANK, for branching and node selection within the SCIP framework on the BN 45 network in terms of average % gap in log-likelihood. More negative values indicate better performance.

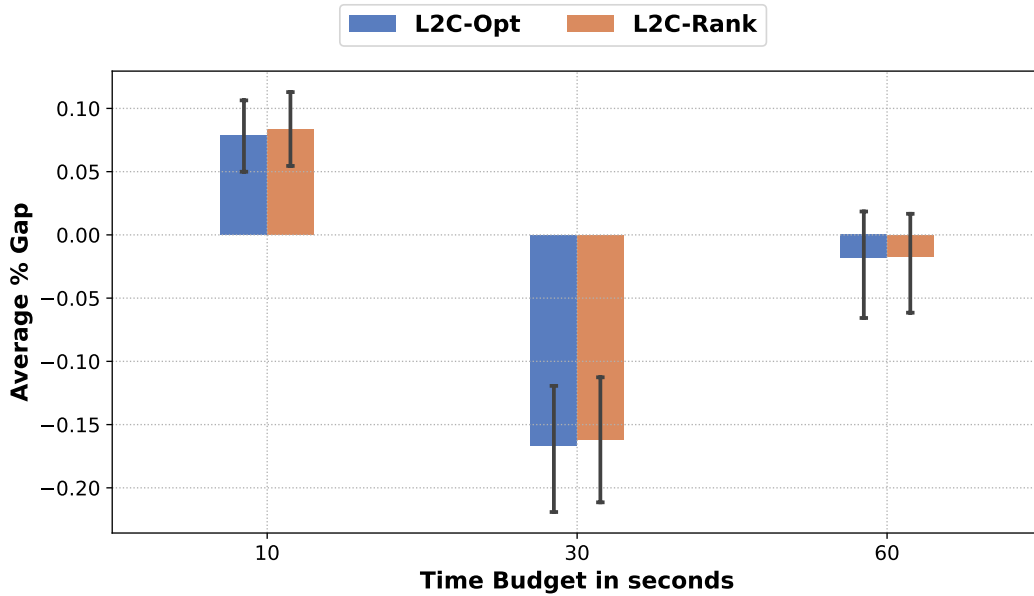


Figure 41: Comparison of SCIP’s default heuristics with our neural strategies, L2C-OPT and L2C-RANK, for branching and node selection within the SCIP framework on the Promedas 68 network in terms of average % gap in log-likelihood. More negative values indicate better performance.

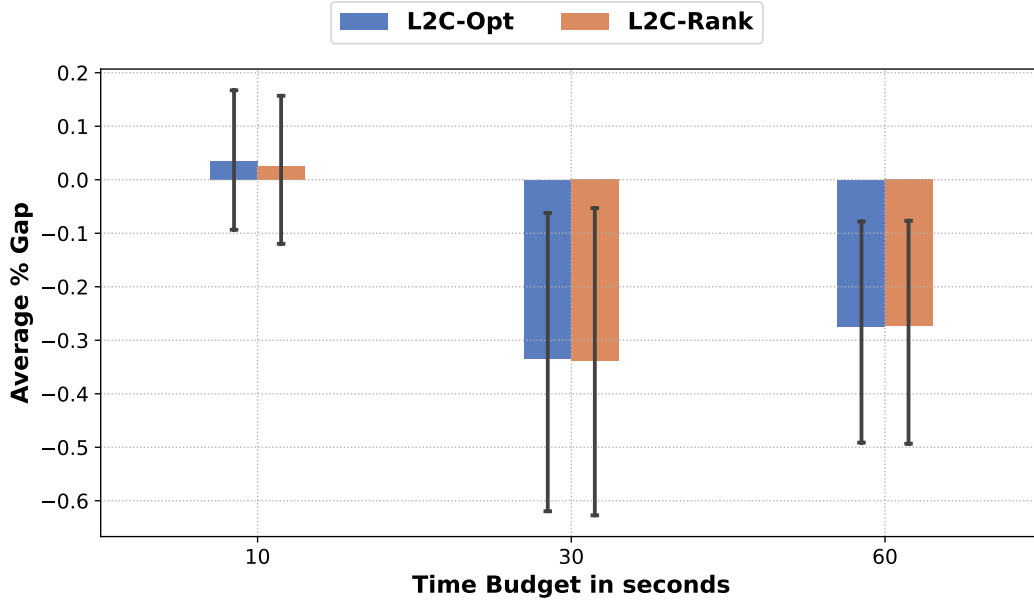


Figure 42: Comparison of SCIP’s default heuristics with our neural strategies, L2C-OPT and L2C-RANK, for branching and node selection within the SCIP framework on the Promedas 60 network in terms of average % gap in log-likelihood. More negative values indicate better performance.

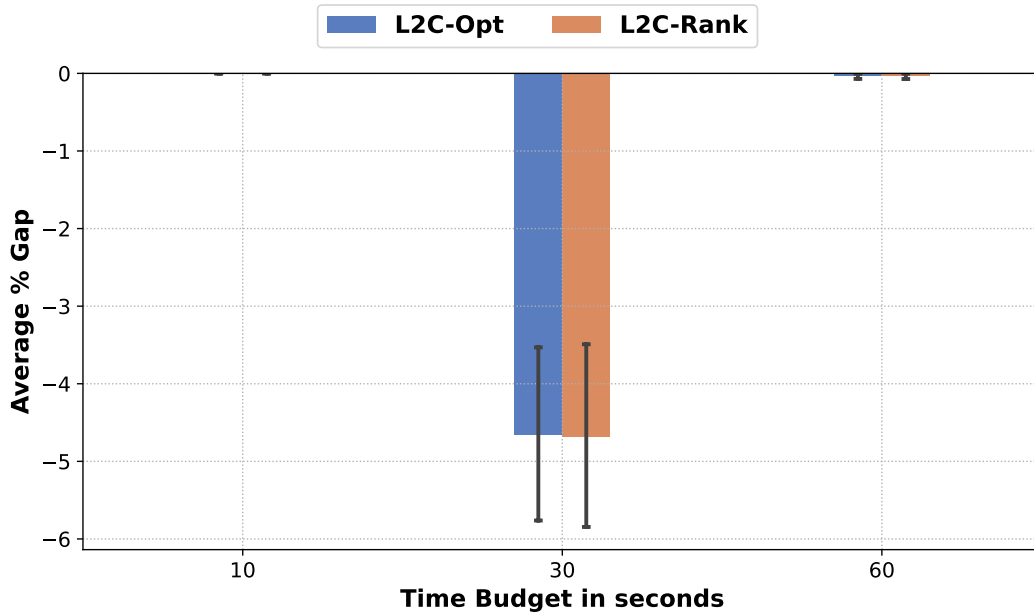


Figure 43: Comparison of SCIP’s default heuristics with our neural strategies, L2C-OPT and L2C-RANK, for branching and node selection within the SCIP framework on the BN 30 network in terms of average % gap in log-likelihood. More negative values indicate better performance.

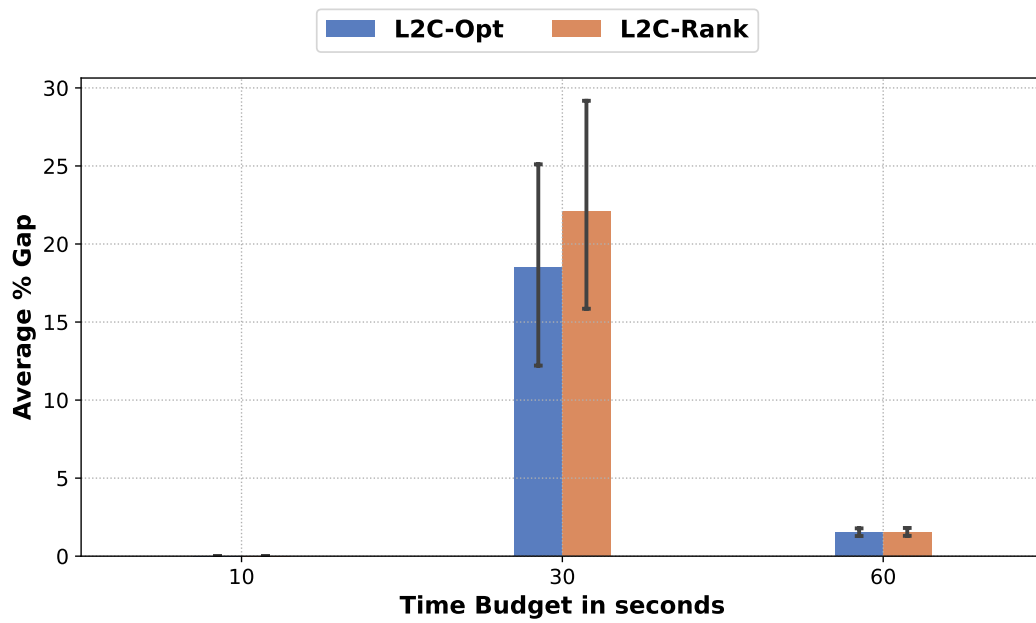


Figure 44: Comparison of SCIP’s default heuristics with our neural strategies, L2C-OPT and L2C-RANK, for branching and node selection within the SCIP framework on the BN 32 network in terms of average % gap in log-likelihood. More negative values indicate better performance.

## References

- [1] Suresh Bolusani, Mathieu Besançon, Ksenia Bestuzheva, Antonia Chmiela, João Dionísio, Tim Donkiewicz, Jasper van Doornmalen, Leon Eifler, Mohammed Ghannam, Ambros Gleixner, Christoph Graczyk, Katrin Halbig, Ivo Hedtke, Alexander Hoen, Christopher Hojny, Rolf van der Hulst, Dominik Kamp, Thorsten Koch, Kevin Kofler, Jurgen Lentz, Julian Manns, Gioni Mexi, Erik Mühmer, Marc E. Pfetsch, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Mark Turner, Stefan Vigerske, Dieter Weninger, and Lixing Xu. The SCIP Optimization Suite 9.0. Technical report, Optimization Online, 2024. URL <https://optimization-online.org/2024/02/the-scip-optimization-suite-9-0/>.
- [2] Lars Otten and Rina Dechter. A case study in complexity estimation: Towards parallel branch-and-bound over graphical models. In Nando de Freitas and Kevin P. Murphy, editors, *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence, Catalina Island, CA, USA, August 14-18, 2012*, pages 665–674. AUA Press, 2012. URL [https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article\\_id=2327&proceeding\\_id=28](https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=2327&proceeding_id=28).
- [3] Lars Otten. Daoopt: Sequential and distributed and/or branch and bound for mpe problems, 2012. URL <https://github.com/lotten/daoopt>.
- [4] Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. Exact combinatorial optimization with graph convolutional neural networks. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d’Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 15554–15566, 2019. URL <https://proceedings.neurips.cc/paper/2019/hash/d14c2267d848abeb81fd590f371d39bd-Abstract.html>.
- [5] Yunzhuang Shen, Yuan Sun, Andrew Eberhard, and Xiaodong Li. Learning primal heuristics for mixed integer programs. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2021. doi: 10.1109/IJCNN52387.2021.9533651.
- [6] Tobias Achterberg, Timo Berthold, and Gregor Hendel. *Rounding and Propagation Heuristics for Mixed Integer Programming*, page 71–76. Springer Berlin Heidelberg, 2012. ISBN 9783642292101. doi: 10.1007/978-3-642-29210-1\_12. URL [http://dx.doi.org/10.1007/978-3-642-29210-1\\_12](http://dx.doi.org/10.1007/978-3-642-29210-1_12).
- [7] Tobias Achterberg and Timo Berthold. Hybrid branching. In *Proceedings of the 6th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, CPAIOR ’09*, page 309–311, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 9783642019289. doi: 10.1007/978-3-642-01929-6\_23. URL [https://doi.org/10.1007/978-3-642-01929-6\\_23](https://doi.org/10.1007/978-3-642-01929-6_23).