Learning to Condition: A Neural Heuristic for Scalable MPE Inference

Brij Malhotra
Department of Computer Science
The University of Texas at Dallas
brijgulsharan.malhotra@utdallas.edu

Shivvrat Arya
Department of Computer Science
New Jersey Institute of Technology
shivvrat.arya@njit.edu

Tahrima Rahman
Department of Computer Science
The University of Texas at Dallas
tahrima.rahman@utdallas.edu

Vibhav Gogate
Department of Computer Science
The University of Texas at Dallas
vibhav.gogate@utdallas.edu

Abstract

We introduce *learning to condition* (L2C), a scalable, data-driven framework for accelerating Most Probable Explanation (MPE) inference in Probabilistic Graphical Models (PGMs), a fundamentally intractable problem. L2C trains a neural network to score variable-value assignments based on their utility for conditioning, given observed evidence. To facilitate supervised learning, we develop a scalable data generation pipeline that extracts training signals from the search traces of existing MPE solvers. The trained network serves as a heuristic that integrates with search algorithms, acting as a conditioning strategy prior to exact inference or as a branching and node selection policy within branch-and-bound solvers. We evaluate L2C on challenging MPE queries involving high-treewidth PGMs. Experiments show that our learned heuristic significantly reduces the search space while maintaining or improving solution quality over state-of-the-art methods.

1 Introduction

Probabilistic Graphical Models (PGMs) [1], such as Bayesian Networks and Markov Networks, efficiently encode joint probability distributions over a large set of random variables, enabling structured reasoning under uncertainty. A fundamental inference task in these models is the Most Probable Explanation (MPE) query, where the goal is to find the most likely assignment of values to unobserved variables given observed evidence.

Answering MPE queries is computationally intractable in general due to their NP-hardness, and becomes particularly challenging as model complexity and scale increase. Classical exact methods such as AND/OR search [2, 3] and integer linear programming (ILP) [1]guarantee optimality but are prohibitively expensive for large instances. Approximate methods offer scalability but often compromise on solution quality or consistency, especially in domains that demand high-precision inference.

Conditioning on a variable subset—fixing variables to specific values—is a common strategy for improving the tractability of inference by simplifying the underlying problem structure. This reduction in complexity can significantly shrink the search space and improve solver performance. This principle underlies classical techniques like cutset conditioning [4] and recursive conditioning [5], as well as strong branching heuristics [6, 7] in ILP solvers. A related concept in statistical physics is *decimation* [8–12], which iteratively fixes high-impact variables to reduce the search space in message-passing algorithms such as Survey Propagation, progressively simplifying the problem instance. However, the

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

effectiveness of these methods is highly sensitive to the choice and ordering of conditioned variables; poor decisions can lead to significant degradation in solution quality.

This leads to a fundamental question: *which variables should be fixed, to which values, and in what order, to simplify inference while preserving the optimal solution?* Fixing variables incorrectly can irrevocably exclude the true optimal solution, whereas deferring all conditioning preserves completeness but forfeits the computational benefits of simplification. Ideally, conditioning decisions should be both *safe*−i.e., aligned with optimal solutions and *useful*−i.e., significantly reduce computational effort.

In principle, access to the complete set of MPE solutions for a given instance would enable the identification of such beneficial assignments. For instance, if a variable assumes the same value across all optimal solutions, fixing it would preserve optimality and likely simplify the subsequent problem. In practice, however, enumerating all MPE solutions is infeasible [13], and exhaustively evaluating the impact of every variable-value pair on solver performance is prohibitively expensive. Consequently, existing methods rely on handcrafted heuristics [14, 15] or structure-based metrics [16] that often lack generalization.

In this work, we introduce *learning to condition* (L2C), a *data-driven* approach to rank variable-value assignments based on their utility for conditioning. The goal is to train a neural network to identify assignments that strike a balance between two objectives: preserving access to optimal solutions and simplifying the inference process. To achieve this, we train an attention-based neural network that assigns two scores to each variable-value assignment: an *optimality score*, estimating the likelihood of the assignment's presence in an optimal solution, and a *simplification score*, measuring the extent to which fixing the assignment reduces solver effort.

We obtain supervision for these scores via a scalable data generation pipeline. For each training instance, we solve the MPE query once using an oracle and treat the resulting solution as a proxy for the set of optimal solutions. Variable-value pairs present in the solution are labeled as positive examples for the optimality head. To supervise the simplification head, we fix individual variables to their optimal values, re-solve the query, and record solver statistics such as runtime and the number of explored nodes, providing a quantitative measure of the reduction in inference complexity.

A key insight of our method is that solution ambiguity modulates conditioning risk. If a variable takes the same value across all optimal solutions, fixing it is highly effective but also risky if the estimate is incorrect, as it could eliminate the true MPE solution. Conversely, when a variable's values are more evenly distributed, conditioning poses less risk. Even if the model slightly misestimates the correct value, the solution space is more tolerant, and optimality may still be preserved. Our model implicitly learns to navigate this trade-off by generating soft scores that reflect both expected utility and uncertainty, shaped through diverse training examples.

At inference time, the model assigns scores to all variable-value pairs in the query set for the given instance and conditions on those with the highest scores. This strategy incrementally simplifies the problem while avoiding premature elimination of the optimal solution. Additionally, the learned model can also be integrated into branch-and-bound solvers as node and variable selection heuristics.

Contributions. This paper introduces a neural network-based conditioning strategy for MPE inference in PGMs, with the following key contributions:

- We formalize *learning to condition* (L2C) as a scoring problem over variable-value pairs that jointly optimizes for solution preservation and inference tractability.
- We design a data-efficient supervision strategy that generates labels using oracle solutions and solver statistics, avoiding the need for exhaustive MPE enumeration.
- We introduce an attention-based architecture that generalizes across instances and yields informative optimality and simplification scores.
- We demonstrate that our method improves both inference efficiency and solution quality over classical heuristics across a range of benchmark PGMs.

Our results show that L2C enables scalable, learned conditioning decisions that adapt to instance structure, outperforming traditional heuristics in both speed and accuracy. Our implementation, solver integrations, and experiment scripts are publicly available at, `https://github.com/brijml/L2C`.

2 Background

We assume, w.l.o.g., that all random variables are binary and take values in $\{0, 1\}$. Individual variables are denoted by uppercase letters (e.g., X), and their assignments by corresponding lowercase letters (e.g., x). Sets of variables are denoted by bold uppercase letters (e.g., $\mathbf{X}$), and assignments to those sets are written in bold lowercase (e.g., $\mathbf{x}$). Given a full assignment $\mathbf{x}$ to a set $\mathbf{X}$, and a subset $\mathbf{Y} \subseteq \mathbf{X}$, we use $\mathbf{x}_\mathbf{Y}$ to denote the projection of $\mathbf{x}$ onto $\mathbf{Y}$.

A **probabilistic graphical model** (PGM) $\mathcal{M} = \langle \mathbf{X}, \mathbf{F}, G \rangle$ is defined over a set of variables $\mathbf{X}$, a collection of log-potentials $\mathbf{F} = \{f_1, \ldots, f_m\}$, and an undirected primal graph $G = (\mathcal{V}, \mathcal{E})$. Each log-potential $f_i \in \mathbf{F}$ is defined over a subset $\mathbf{S}(f_i) \subseteq \mathbf{X}$, known as its scope. The vertex set $\mathcal{V}$ contains one node for each variable in $\mathbf{X}$, and an edge $(\mathcal{V}_a, \mathcal{V}_b) \in \mathcal{E}$ is added whenever X_a and X_b co-occur in the scope of some f_i. The model defines the following joint probability distribution:

$$P_\mathcal{M}(\mathbf{x}) \propto \exp\left(\sum_{f \in \mathbf{F}} f(\mathbf{x}_{\mathbf{S}(f)})\right)$$

We focus on the **most probable explanation (MPE)** task: given observed evidence variables $\mathbf{E} \subset \mathbf{X}$ with assignment $\mathbf{e}$, find the most likely assignment for query variables $\mathbf{Q} = \mathbf{X} \setminus \mathbf{E}$. Formally, this is:

$$\text{MPE}(\mathbf{Q}, \mathbf{e}) = \arg\max_{\mathbf{q}} P_\mathcal{M}(\mathbf{q} \mid \mathbf{e}) = \arg\max_{\mathbf{q}} \sum_{f \in \mathbf{F}} f((\mathbf{q}, \mathbf{e})_{\mathbf{S}(f)})$$

This problem is NP-hard in general and remains intractable for numerous model families [17–21].

Exact MPE algorithms include bucket elimination [22], which uses variable elimination via local reparameterization, and Mixed Integer Programming (MIP) encodings solved with branch-and-bound solvers like SCIP [23] that combine LP relaxations with systematic search. Such solvers are often anytime, providing progressively better solutions and bounds, with optimality certificates upon convergence. Nevertheless, for many real-world PGMs—particularly those with high treewidth—exact inference remains computationally infeasible.

Conditioning is a classical strategy for simplifying inference by fixing a subset of variables. For MPE, assigning values $\mathbf{q}_d$ to a conditioning set $\mathbf{Q}_d \subseteq \mathbf{Q}$ reduces the problem to solving MPE for the remaining variables $\mathbf{Q}_0 = \mathbf{Q} \setminus \mathbf{Q}_d$ given evidence $\mathbf{e} \cup \mathbf{q}_d$. The resulting residual solution $\mathbf{q}_0^*$ and $\mathbf{q}_d$ form the complete MPE solution. Conditioning is fundamental to exact methods like cutset conditioning [4], recursive conditioning [5], and search algorithms such as Branch and Bound (B&B) and AND/OR Branch and Bound (AOBB) [24]. Conditioning also informs heuristic methods such as decimation strategies inspired by statistical physics [9, 10, 25, 26].

Branch and Bound (B&B) performs a depth-first or best-first search over a decision tree, recursively partitioning the assignment space and pruning subproblems using lower bounds from LP relaxations or mini-bucket elimination [27]; its efficiency hinges on branching heuristics and bound quality. **AND/OR Branch and Bound (AOBB)** [24] improves upon B&B by exploiting graphical model structure through a *pseudo tree* and an AND/OR search graph. This graph uses OR nodes for variable choices and AND nodes for decomposition points, enabling independent search on conditionally independent components formed by assignments. AOBB further avoids redundant computations by merging nodes with identical separator contexts. Consequently, AOBB can prune substantial portions of the search space, often achieving exponential savings over standard B&B.

Graph-based methods like Mini-Bucket Elimination [27] generate bounds (e.g., upper bounds for MPE) that accelerate B&B and AOBB through earlier pruning. Variable and value ordering heuristics further enhance their effectiveness [28–30].

Recent efforts integrate learning into solver decisions. In MIP, *full strong branching* [7, 6] uses one-step look-ahead for selecting branch variables that maximize bound improvement; neural networks trained via imitation learning can replicate this [31, 32]. Other machine learning approaches include using SVMs to identify optimal decision variables [33] and NNs for effective node-selection strategies [34]. Recent neural approaches to MPE inference have concentrated on optimizing relaxed likelihood objectives [35–37]. While these methods eliminate the need for supervision, they provide no guarantees of optimality, as they rely on continuous relaxations of the MPE objective.

Our work advances *conditioning-based simplification* by learning a scoring function that ranks variable-value assignments according to their utility in reducing inference complexity while preserving optimality.

3 Learning to Condition (L2C)

We propose a neural approach for ranking variable-value assignments based on their utility in simplifying MPE inference while preserving optimality. This task, which we call **Learning to Condition (L2C)**, balances two goals: minimizing inference cost and avoiding exclusion of optimal solutions. Our model learns two scores per assignment: one measuring consistency with optimal solutions and another estimating simplification benefit. These data-driven scores guide the selection of high-confidence assignments and can also be used to steer node selection in B&B solvers. Next, we describe the L2C pipeline, including data generation, neural architecture, training objectives, ranking mechanism, and integration with inference procedures.

3.1 Data collection

We construct a dataset of MPE queries and solver outcomes to train the L2C model to score variable-value assignments based on two criteria: their consistency with optimal solutions and their ability to simplify inference when fixed. Because enumerating all optimal solutions or exhaustively evaluating all assignments is computationally infeasible, we collect targeted supervision using selective oracle queries under randomized input conditions.

The full data collection procedure is described in Algorithm 1. Given a probabilistic graphical model $\mathcal{M} = \langle \mathbf{X}, \mathbf{F}, G \rangle$, we repeatedly sample full assignments $\mathbf{x} \sim P_{\mathcal{M}}$ and randomly partition the variable set into disjoint subsets: a query set $\mathbf{Q} \subset \mathbf{X}$ of size $qr \cdot |\mathbf{X}|$, and an evidence set $\mathbf{E} = \mathbf{X} \setminus \mathbf{Q}$. The evidence assignment $\mathbf{e}$ is formed by projecting $\mathbf{x}$ onto $\mathbf{E}$, and an MPE solver is invoked to compute the optimal solution $\mathbf{q}^* = \arg\max_{\mathbf{q}} P_{\mathcal{M}}(\mathbf{q} \mid \mathbf{e})$. The resulting assignment, along with runtime and number of nodes explored, is stored as supervision.

To evaluate the impact of variable-value assignments on inference, we randomly sample a subset $\mathbf{C} \subset \mathbf{Q}$ of size at most $c_{\max}$, and for each $C \in \mathbf{C}$, consider both binary assignments $C = 0$ and $C = 1$. For each assignment, we augment the evidence to form $\mathbf{e}' = \mathbf{e} \cup \{C = c\}$, solve the resulting MPE query, and record solver statistics such as runtime, number of nodes explored, and objective value. These evaluations provide supervision for both the simplification and optimality scoring heads.

Evaluating all possible variable-value assignments for each instance is computationally prohibitive, especially for large models. To manage this cost, we introduce a sampling parameter $c_{\max}$ that bounds the number of variables considered for conditioning per instance. This allows us to gather informative supervision while keeping the number of additional MPE queries tractable. In practice, $c_{\max}$ can be chosen based on the model size, the characteristics of the solver, and the available compute budget.

When the solver fails to return a result within the allocated time budget B, we log surrogate statistics such as LP-bound improvements or relaxed objective scores. Although these surrogates may be noisy, they increase coverage across instances and improve the generalization ability.

Algorithm 1: Data Collection for L2C with Conditioning Assignments

Input: PGM $\mathcal{M} = \langle \mathbf{X}, \mathbf{F}, G \rangle$, Query Ratio qr, Max Conditions $c_{\max}$, Budget B
Output: Dataset DB
Function `SolveMPE`($\mathcal{M}, \mathbf{e}, B$):
  Run MPE solver on $\mathcal{M}$ with evidence $\mathbf{e}$ and time bound B
  return rec $= \{\text{assign}, \text{time}, \#\text{nodes}\}$
Initialize $DB \leftarrow \emptyset$
while *not enough samples in* DB **do**
  Sample $\mathbf{x} \sim P_{\mathcal{M}}$
  $\mathbf{Q} \leftarrow$ random subset of $\mathbf{X}$ of size $qr|\mathbf{X}|$
  $\mathbf{E} \leftarrow \mathbf{X} \setminus \mathbf{Q}$
  $\mathbf{e} \leftarrow \mathbf{x}_{\mathbf{E}}$
  rec $\leftarrow$ `SolveMPE`($\mathcal{M}, \mathbf{e}$)
  Store $(\mathbf{e}, \text{rec})$ in DB
  $\mathbf{C} \leftarrow$ random subset of $\mathbf{Q}$ of size $c_{\max}$
  for *each* $C \in \mathbf{C}$ **do**
    for *each value* $c \in \{0, 1\}$ **do**
      $\mathbf{e}' \leftarrow \mathbf{e} \cup \{C = c\}$
      rec$' \leftarrow$ `SolveMPE`($\mathcal{M}, \mathbf{e}'$)
      Store $(\mathbf{e}', \text{rec}')$ in DB
return DB

Each $\mathbf{e}$ and its corresponding optimal assignment $\mathbf{q}^*$ (if returned by the MPE solver) in DB serves as supervision for the optimality head. For simplification, we convert the collected solver statistics into a ranking distribution over a small set of candidate variables $\mathbf{C} \subset \mathbf{Q}$. For each candidate $C \in \mathbf{C}$, we

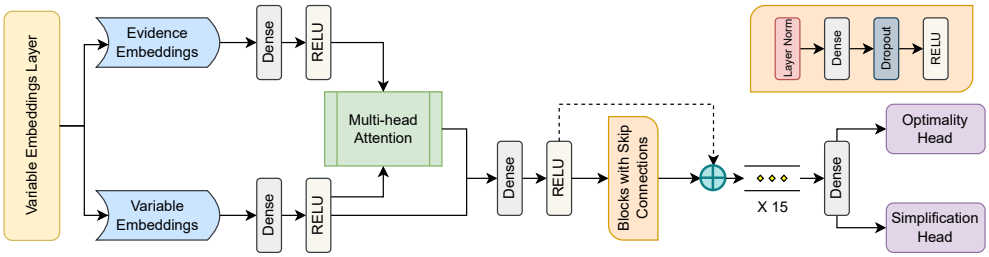


Figure 1: Attention-based architecture for scoring variable-value pairs by their utility in simplifying MPE inference while preserving optimality.

define: $p_C \propto \frac{1}{t_C}$ where t_C is a composite scalar derived from multiple solver statistics recorded for the query conditioned on $C = c$, including runtime, number of search nodes explored, and objective value. These quantities are combined linearly to reflect the overall simplification utility of the assignment. The resulting distribution is normalized using a softmax over inverse scores, assigning higher probabilities to variable-value assignments that are more effective at reducing inference cost.

Thus the resulting dataset contains both exact and surrogate supervision, allowing the model to learn to balance optimality preservation with inference simplification across a range of graphical model instances.

3.2 Neural network architecture

For a partially observed MPE instance (evidence $\mathbf{E} \subset \mathbf{X}$ with assignment $\mathbf{e}$, query variables $\mathbf{Q} = \mathbf{X} \setminus \mathbf{E}$), our model (Figure 1) scores variable-value pairs on their potential for MPE optimality preservation and inference cost reduction. Each pair $(X_i = x_i)$ is represented by an embedding vector [38]. For binary variables, an embedding table of size $2 \times |\mathbf{X}|$ provides distinct embeddings for each variable's possible values. An additional embedding indicates variable status (observed/evidence or unobserved/query), enabling support for arbitrary evidence-query partitions.

For each unobserved variable $Q_i \in \mathbf{Q}$, its two value assignment embeddings query a multi-head attention module [39] using evidence variable embeddings as keys and values, contextualizing each candidate assignment relative to the instance and enabling the model to learn associations between evidence and promising assignments. The output contextualized embeddings, concatenated with their original counterparts, pass through a shared encoder (comprising fully connected layers with ReLU activations, residual connections [40], and dropout [41]). This encoder produces a representation for each variable-value pair, which is then passed to two separate prediction heads.

The *optimality head*, a two-layer MLP with a sigmoid output, estimates an assignment's likelihood of inclusion in the MPE solution. Trained with binary cross-entropy loss against oracle-derived labels, this head learns a relaxed optimality proxy, capturing generalizable patterns across problem instances.

A separate *simplification head*, also a two-layer MLP, outputs unnormalized scores for candidate assignments. These scores, after softmax normalization, are trained via a list-ranking cross-entropy loss [42], using targets derived from solver statistics (Section 3.1) like runtime, search depth, or relaxed objectives. By predicting relative utility instead of absolute complexity reduction, this head produces scores that generalize well and exhibit noise robustness.

Our architecture exhibits permutation invariance to variable ordering and supports arbitrary evidence-query partitions, promoting generalization. Cross-attention facilitates query-specific evidence conditioning, while the dual-head design jointly optimizes solution quality and inference efficiency. This yields a flexible scoring function adept at guiding conditioning during inference.

3.3 Loss functions

We train our model using a multi-task objective [43] that combines two losses: a binary classification loss for predicting whether a variable-value pair is part of an optimal MPE solution, and a list-ranking loss that encourages the model to prioritize assignments that simplify inference. These respectively supervise the network's optimality and simplification heads.

Optimality loss. For each query variable $Q_i \in \mathbf{Q}$ and its possible values $q_i \in \{0, 1\}$, the optimality head predicts $\hat{y}_i^{(q)} \in [0, 1]$, the probability that assignment $Q_i = q$ is in the oracle MPE solution, where $y_i^{(q)} \in \{0, 1\}$ is the true label. The binary cross-entropy loss is applied independently per pair:

$$\mathcal{L}_{\text{opt}} = -\sum_{Q_i \in \mathbf{Q}} \sum_{q \in \{0,1\}} \left(y_i^{(q)} \log \hat{y}_i^{(q)} + (1 - y_i^{(q)}) \log(1 - \hat{y}_i^{(q)}) \right)$$

This formulation supports supervision with multiple optimal assignments, enabling the model to learn from all optimal solutions for a given instance.

Simplification ranking loss. For each instance, the training dataset contains solver statistics for a subset of query variables $\mathbf{C} \subset \mathbf{Q}$, obtained during data collection as described in Section 3.1. For each $C \in \mathbf{C}$, we construct a target probability distribution p_C that reflects the simplification utility of fixing $C = c$, based on a combination of solver statistics such as runtime, number of nodes explored, and objective value. The simplification head generates scores for all variable-value pairs in $\mathbf{Q}$, which are normalized via softmax into a predicted distribution $\hat{p}_C$. We then compute a list-ranking cross-entropy loss [42]:

$$\mathcal{L}_{\text{rank}} = -\sum_{C \in \mathbf{C}} p_C \log \hat{p}_C$$

This loss encourages the model to prioritize assignments that lead to greater reductions in inference cost. Since supervision is available only for a subset of $\mathbf{Q}$, we apply a binary mask to restrict the loss computation to the subset $\mathbf{C}$ for which solver statistics were recorded during training. This ensures that only observed variable-value pairs influence parameter updates, while predictions for the rest of $\mathbf{Q}$ remain unconstrained.

Joint objective and masking. The total loss combines these components as a weighted sum:

$$\mathcal{L} = \lambda_{\text{opt}} \cdot \mathcal{L}_{\text{opt}} + \lambda_{\text{rank}} \cdot \mathcal{L}_{\text{rank}}$$

Here, λ_{opt} and λ_{rank} balance the objectives' relative importance. We train using both exact and surrogate datasets. If supervision is available for only one head (optimality or ranking), we apply its loss and mask the other, preventing gradient updates from missing targets. This joint objective allows the model to balance accuracy with inference efficiency by leveraging both exact and approximate supervision.

Algorithm 2: Beam Search for L2C

Input: PGM $\mathcal{M}$, Scoring function $\mathcal{F}$, Initial evidence $\mathbf{E}$, Time limit tl, Max depth $\mathcal{D}_{\max}$, Beam width W
Output: Approximate MPE solution
Init. beam $B \leftarrow \{(0, \mathbf{E})\}$ `//(score,evid)`
for $d = 1$ *to* $\mathcal{D}_{\max}$ **do**
  Initialize candidate list $C \leftarrow \emptyset$
  foreach $(s, \mathbf{E}_{partial}) \in B$ **do**
    Identify query variables $\mathbf{Q}_{\text{rem}} = \mathbf{X} \setminus \mathbf{E}_{\text{partial}}$
    foreach *unfixed* $Q_i \in \mathbf{Q}_{rem}$ **do**
      foreach *value* $q_i \in \{0, 1\}$ **do**
        Let $\mathbf{E}' \leftarrow \mathbf{E}_{\text{partial}} \cup \{Q_i = q_i\}$
        Let $s' \leftarrow \mathcal{F}(\mathbf{E}')$
        Add $(s', \mathbf{E}')$ to C
  Sort C in descending order by score s'
  Prune C to retain top W elements
  $B \leftarrow C$
Let $(s^*, \mathbf{E}^*) \leftarrow$ TopCandidate(B)
Run $\mathbf{q}^* \leftarrow$ `SolveMPE`$(\mathcal{M}, \mathbf{E}^*, tl)$
return $\mathbf{q}^*$

3.4 Inference-time strategies

At inference time, the model guides variable-value assignment decisions that simplify the MPE problem before the solver is invoked. Given a probabilistic graphical model $\mathcal{M} = (\mathbf{X}, \mathcal{F})$, an initial evidence set $\mathbf{E} \subset \mathbf{X}$, and assignment $\mathbf{e}$, the model takes the current evidence as input and produces scores for each remaining query variable-value pair $(Q_i = q_i)$, where $Q_i \in \mathcal{V} \setminus \mathbf{E}$ and $q_i \in \{0, 1\}$.

The *optimality head* outputs confidence $\hat{y}_i^{(q)} \in [0, 1]$ for an assignment's MPE consistency; the *simplification head* provides a score $s_i^{(q)} \in \mathbb{R}$ for its inference simplification utility. These scores are combined to select conditioning assignments via two strategies:

Greedy conditioning. This strategy iteratively filters assignments by an optimality threshold τ, selects the one with the highest simplification score among confident candidates, and adds it to evidence. This repeats for a fixed number of steps or until no confident options remain, before passing the simplified MPE query to a solver.

Beam search. To explore multiple conditioning sequences, beam search (Algorithm 2) maintains W partial assignment sequences. At each step $d \leq \mathcal{D}_{\max}$, sequences are expanded with assignments scored as in greedy conditioning. The best final sequence provides evidence $\mathbf{E}^*$ for the MPE solver.

Final solution. Let $\mathbf{E}^*$ denote the final evidence set selected by either strategy. The residual MPE query is defined over $\mathbf{Q}_{\text{residual}} = \mathcal{V} \setminus \mathbf{E}^*$. We call an MPE solver (exact or anytime) to compute:

$$\mathbf{q}^*_{\text{residual}} = \arg\max_{\mathbf{q}} P_{\mathcal{M}}(\mathbf{q} \mid \mathbf{e}^*)$$

and return the complete solution $\mathbf{q}^* = \mathbf{q}^*_{\text{residual}} \cup \mathbf{e}^*$.

This framework uses learned assignments to restructure the MPE problem, reducing solver complexity while preserving correctness and enabling time-bounded inference.

NN-Guided branch-and-bound search Our neural model also enhances branch-and-bound (B&B) solvers. It guides branching by selecting variables predicted to simplify the subproblem (using scores akin to greedy conditioning) and uses optimality head predictions to tighten lower bounds. These actions accelerate B&B convergence while maintaining optimality guarantees.

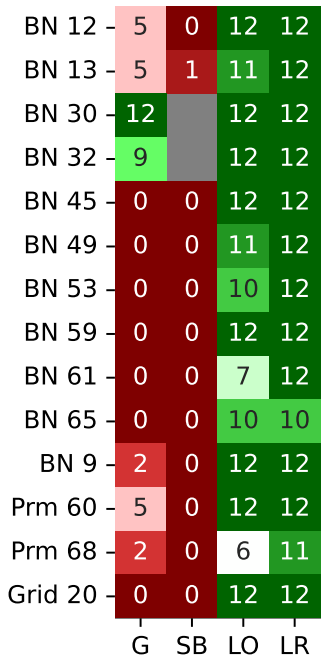
Learning to Condition (L2C) transforms the traditionally static variable selection problem into a learned, instance-specific decision process. Compatible with both exact and approximate solvers, L2C generalizes across problem instances and enables principled trade-offs between solution quality and computational cost. We next empirically evaluate its performance across a diverse set of benchmarks and graphical models.

4 Experiments

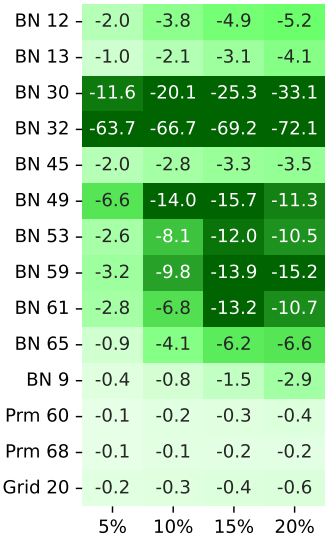
In this section, we evaluate our neural strategies, L2C-OPT (using only the optimality head for scoring) and L2C-RANK (using the scoring procedure from Section 3.4). We benchmark these strategies for two tasks: (1) against standard conditioning heuristics, and (2) as branching and node selection methods within B&B solvers. We begin by describing our experimental framework, including the PGMs, baseline methods, evaluation metrics, and NN architecture.

4.1 Graphical models

We evaluated our method and baselines on 14 high-treewidth binary probabilistic graphical models (PGMs) from UAI inference competitions [44, 45], featuring 90 to 1444 variables and up to 1444 factors. Using Gibbs sampling [1], we generated 12,000 training, 1,000 test, and 1,000 validation examples per model. All MPE instances used 75% of the PGM's variables as query variables. We construct the supervised dataset following the procedure outlined in Section 3.1. During data generation, QR is set to 0.75 and $c_{\max}$ is set to 10% of the total number of variables in each PGM.



(a) Wins vs. oracle by time limit for all methods (G: Graph, SB: Full Strong Branching, LO: L2C-OPT, LR: L2C-RANK).



(b) L2C-RANK vs. oracle $\mathcal{LL}$ gap by $\mathcal{D}_{max}$ (cols) and graphical model (rows).

Figure 2: Neural vs. baseline methods for greedy conditioning (SCIP oracle). (a) Win counts; (b) Log-likelihood gaps. Color: Darker green = stronger, darker red = weaker performance. Grey: Timeouts (30s).

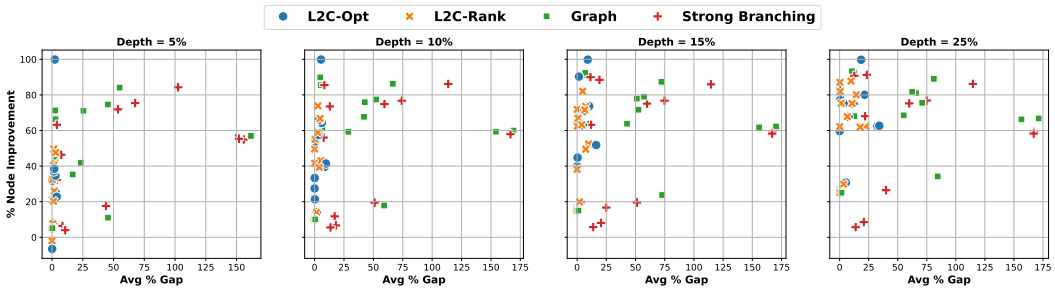


Figure 3: Greedy conditioning methods (AOBB oracle): Average solution gap (x-axis) versus % node reduction (y-axis). Each subfigure denotes a fixed decision count.

4.2 Experimental setup and methods

The neural networks in L2C-OPT and L2C-RANK use 256-dimensional embeddings, two multi-head attention layers, and 15 skip-connection blocks. Dense layers have 512 units with 0.1 dropout [41] and ReLU activations. We trained the models using Adam [46] with a learning rate of 8×10^{-4}, an exponential decay rate of 0.97, a batch size of 128, and early stopping after 5 stagnant epochs (maximum 50 epochs). All models were implemented in PyTorch [47] and executed on an NVIDIA A40 GPU. Appendix B provides further hyper-parameter details.

Baseline methods: We compare our approach with two heuristic methods: The **graph-based heuristic** [28–30] prioritizes variables by degree, selecting them in descending order after removing evidence nodes. This max-degree strategy assigns each variable its most likely value, estimated via Gibbs Sampling [1, 48] over the PGM. The **full strong branching heuristic** [7], a classical technique for enhancing branch-and-bound efficiency, formulates the MPE query as an ILP [1]. It then scores variable-value pairs by their impact on search space pruning.

Oracle: We use two oracles on unconditioned queries and as final solvers post-conditioning: the SCIP solver [23], and the AND/OR Branch and Bound (AOBB) solver (per Otten and Dechter [24] and implemented by Otten [3]). MPE queries are encoded as Integer Linear Programs (ILPs) [1] for SCIP, which applies its branch-and-bound algorithm subject to a 60-second time limit per query.

Evaluation criteria: The competing approaches were evaluated based on two criteria: log-likelihood scores and runtime. The log-likelihood scores, calculated as $\ln p_{\mathcal{M}}(\mathbf{e}, \mathbf{q})$, assesses the quality of the solution in terms of optimality. The runtime measures the efficiency of the conditioning process by indicating how quickly each method solves the problem.

4.3 Empirical evaluations

4.3.1 Greedy conditioning

We evaluate our methods and baselines as scoring functions ($\mathcal{F}$) in a greedy conditioning framework (Section 3.4), selecting up to 25% of query variables.[1] For each PGM, we test 12 configurations: four conditioning depths (5%, 10%, 15%, 25% of query variables) combined with three oracle time budgets (10s, 30s, 60s). Post-conditioning, SCIP attempts to solve the residual problem within the allotted budget.

Figure 2a (heatmap) quantifies for each conditioning strategy the number of the 12 configurations where it enabled the SCIP oracle to achieve a higher average log-likelihood ($\mathcal{LL}$) score than solving the original, non-conditioned query. The oracle's time budget was identical for both conditioned and non-conditioned queries.

The results (Figure 2a) show that both L2C-OPT (LO) and L2C-RANK (LR) consistently enhance oracle performance over direct solving of the original query. L2C-RANK performs best, frequently

[1]To ensure a fair comparison, as the max-degree baseline supports only a beam size of 1, we use greedy conditioning for neural and strong branching methods. Appendix E provides beam search results.

surpassing the non-conditioned oracle across all 12 configurations. Conversely, full strong branching (SB) and the graph-based heuristic (G) only occasionally improve $\mathcal{LL}$ scores.

Figure 2b presents the average percentage $\mathcal{LL}$ gap: $\frac{1}{N}\sum_{i=1}^{N}(\mathcal{LL}_S^{(i)} - \mathcal{LL}_D^{(i)})/(|\mathcal{LL}_S^{(i)}|) \times 100$, where $\mathcal{LL}_S^{(i)}$ and $\mathcal{LL}_D^{(i)}$ are the oracle's log-likelihoods for instance i without and with conditioning by L2C-RANK, respectively. Rows denote PGMs, and columns indicate conditioning depth (percentage of query variables conditioned). The oracle's time budget is fixed at 30s for both approaches.

Predominantly negative (green) values in Figure 2b indicate that L2C-RANK conditioning helps the oracle find superior solutions within the 30s budget. The gap becomes increasingly negative with more conditioning decisions (i.e., greater conditioning depth), demonstrating our method's scalability. This improvement occurs because conditioning simplifies the query by instantiating additional variables, thus presenting an easier query to the oracle.

Assisting AOBB via conditioning: We also evaluated our neural heuristic with the AND/OR Branch and Bound (AOBB) solver [3] as an alternative oracle.

Figure 3 plots the percentage reduction in AOBB's processed node count (y-axis, proxy for effort reduction) against the average $\mathcal{LL}$ gap from the exact MPE solution (x-axis).[2] Points near the y-axis indicate high solution quality with reduced effort; greater x-values imply more quality degradation.

Our conditioning strategies, as shown in Figure 3, yield near-optimal solutions while substantially reducing AOBB's processed node count. In contrast, traditional baselines exhibit larger solution gaps with increased conditioning depth due to increasingly suboptimal decisions. Our methods consistently maintain high solution quality (most points near the y-axis) and more effectively accelerate AOBB by shrinking the search space, particularly with deeper conditioning.

Consequently, L2C enables efficient resolution of queries that are otherwise challenging for the oracle, especially for large-treewidth models, frequently yielding substantially better solutions under identical time constraints.

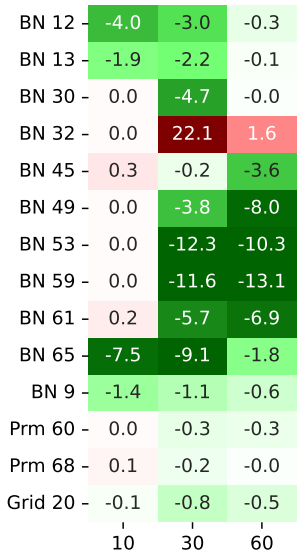


Figure 4: Heuristic comparison across datasets (rows) and time limits (columns); darker green/red indicates better/worse L2C-RANK performance.

4.3.2 NN-Guided branch-and-bound search

We also employ our neural network outputs to guide branch-and-bound search, using them as branching rules [32] and node selection heuristics [34], following the methodology detailed in Section 3.4.

Figure 4 compares our neural-guided heuristics against SCIPs default strategies [23, 49–51], showing the average $\mathcal{LL}$ gap (defined previously) over three time limits and all datasets. The predominantly green cells, particularly dark green ones, in the heatmap signify that our neural heuristics often outperform SCIP's defaults. Furthermore, the few red cells underscore that our approach generally matches or surpasses SCIP's default heuristic performance and delivers better solutions within identical time budgets.

4.3.3 Comparing conditioning methods by per-decision time

Table 1 reports the average time required to make a single conditioning decision across all networks. Both L2C-OPT and L2C-RANK maintain consistent decision times across different network sizes, demonstrating the scalability and practical feasibility of our learned heuristics. In contrast, the graph-based heuristic exhibits increasing latency as network size grows, while strong branching is considerably slower and often infeasible for large models. We imposed a 30-second timeout per decision, under which strong branching failed to return results for the largest networks (BN 30 and

[2]Node and $\mathcal{LL}$ gap reductions for AOBB are relative to its unconditioned execution. AOBB's internal preprocessing and heuristics often enable it to solve MPE queries within the time limit; thus, a reduced node count with a near-zero $\mathcal{LL}$ gap signifies simplified inference without quality loss.

Table 1: Mean ± standard deviation of decision time (s) across networks for each branching strategy. Gray cells indicate cases where the 30 s timeout was reached.

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

BN 32). Overall, these results show that L2C achieves low and predictable decision costs even as model complexity increases.

Summary: Our comprehensive experiments validate the consistent superiority of L2C. We evaluate our approach using two competitive and complementary solver backends: AOBB, a top-performing AND/OR search solver, and SCIP, a state-of-the-art ILP optimizer. This combination encompasses structurally distinct inference paradigms, highlighting the solver-agnostic advantages of learned conditioning. As a conditioning strategy, it empowers SCIP to achieve better solutions within time limits, especially for intractable problems, and allows AOBB to preserve solution quality while drastically reducing search effort (node exploration). Furthermore, employing our neural scores as branching and node selection heuristics enhances SCIP's solution quality under the same time constraints. Importantly, L2C maintains stable decision times across increasing network sizes, underscoring its scalability and suitability for large-scale inference. Collectively, these results affirm our method's effectiveness both as a conditioning strategy, improving solution quality and efficiency and as an effective heuristic guiding branch-and-bound solvers.

5 Conclusion

We introduced *Learning to Condition* (L2C), a neural approach leveraging solver search traces to accelerate Most Probable Explanation (MPE) inference in Probabilistic Graphical Models (PGMs). L2C learns to identify variable assignments that effectively balance inference simplification with optimality preservation, superseding handcrafted heuristics. It serves either as a pre-processor simplifying problems for exact solvers or as an internal policy guiding branching and variable selection within search algorithms. Evaluations on high-treewidth models show L2C substantially curtails the search space, achieving solution quality comparable or superior to baselines and thus providing a more efficient path to MPE solutions.

Limitations and future work: The current limitations lie in scalability, solver signal diversity, and generalization to unseen model structures. The method is tailored to a fixed PGM and depends on solver-derived supervision, which is costly to obtain at scale. Extending L2C to other solver families, such as MAXSAT, local search, or soft arc-consistency solvers, would necessitate substantial modifications to the architecture, data collection pipeline, and learning objectives.

Future work will focus on scaling L2C to handle larger models with millions of variables and factors, while employing more efficient and bootstrapped data collection techniques to reduce oracle dependence. We will also integrate richer solver signals (e.g., branch-and-cut statistics) to enhance pruning and design neural bounding mechanisms that extend learned guidance beyond conditioning. Finally, learned variable orderings will be explored for novel neural bounding strategies, extending guidance beyond conditioning to further boost inference performance.

Acknowledgments and Disclosure of Funding

This work was supported in part by the DARPA CODORD program under contract number HR00112590089, the DARPA Assured Neuro Symbolic Learning and Reasoning (ANSR) Program under contract number HR001122S0039, the National Science Foundation grant IIS-1652835, and the AFOSR award FA9550-23-1-0239.

References

[1] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. Adaptive computation and machine learning. MIT Press, 2009. ISBN 9780262013192.

[2] Radu Marinescu and Rina Dechter. And/or branch-and-bound for graphical models. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, IJCAI'05, page 224229, San Francisco, CA, USA, 2005. Morgan Kaufmann Publishers Inc.

[3] Lars Otten. Daoopt: Sequential and distributed and/or branch and bound for mpe problems, 2012. URL `https://github.com/lotten/daoopt`.

[4] Judea Pearl and Rina Dechter. Identifying independence in bayesian networks. *Networks*, 20 (5):507–534, 1990. doi: 10.1002/net.3230200506.

[5] Adnan Darwiche. Recursive conditioning. *Artificial Intelligence*, 126(1):5–41, 2001. ISSN 0004-3702. doi: 10.1016/S0004-3702(00)00069-2.

[6] Santanu S. Dey, Yatharth Dubey, Marco Molinaro, and Prachi Shah. A theoretical and computational analysis of full strong-branching. *Math. Program.*, 205(12):303336, 2023. ISSN 0025-5610. doi: 10.1007/s10107-023-01977-x. URL `https://doi.org/10.1007/s10107-023-01977-x`.

[7] D. Applegate, R. Bixby, V. Chvatal, and B. Cook. Finding cuts in the tsp (a preliminary report). Technical report, 1995.

[8] M Mézard, G Parisi, and R Zecchina. Analytic and algorithmic solution of random satisfiability problems. *Science*, 297(5582):812–815, 2002.

[9] A. Braunstein, M. Mézard, and R. Zecchina. Survey propagation: An algorithm for satisfiability. *Random Struct. Algorithms*, 27(2):201226, 2005. ISSN 1042-9832.

[10] Lukas Kroc, Ashish Sabharwal, and Bart Selman. Message-passing and local heuristics as decimation strategies for satisfiability. In *Proceedings of the 2009 ACM symposium on Applied Computing*, pages 1408–1414, Honolulu Hawaii, 2009. ACM. ISBN 978-1-60558-166-8. doi: 10.1145/1529282.1529596. URL `https://dl.acm.org/doi/10.1145/1529282.1529596`. TLDR: The results reveal that once the authors resolve convergence issues, BP itself can solve fairly hard random k-SAT formulas through decimation; the gap between BP and SP narrows down quickly as k increases, and the hardness of the decimated formulas is explored.

[11] Keki Burjorjee. Explaining adaptation in genetic algorithms with uniform crossover: The hyperclimbing hypothesis. In *Proceedings of the 14th annual conference companion on Genetic and evolutionary computation*, pages 1461–1462, Philadelphia Pennsylvania USA, 2012. ACM. ISBN 978-1-4503-1178-6. doi: 10.1145/2330784.2330991. URL `https://dl.acm.org/doi/10.1145/2330784.2330991`.

[12] Lukas Kroc. *Probabilistic techniques for constraint satisfaction problems*. PhD thesis, USA, 2009. AAI3376595.

[13] Radu Marinescu and Rina Dechter. Counting the optimal solutions in graphical models. 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 12091–12101, 2019. URL `https://proceedings.neurips.cc/paper/2019/hash/fc2e6a440b94f64831840137698021e1-Abstract.html`.

[14] David Poole and Nevin Lianwen Zhang. Exploiting contextual independence in probabilistic inference. *J. Artif. Int. Res.*, 18(1):263313, 2003. ISSN 1076-9757.

[15] A Fridman. Mixed markov models. *Proceedings of the National Academy of Sciences*, 100(14): 8092–8096, 2003.

[16] Uffe Kjærulff. Triangulation of graph — algorithms giving small total state space. Technical Report R90-09, Aalborg University, Denmark, 1990.

[17] Gregory F. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, 42(2-3):393–405, 1990. ISSN 00043702. doi: 10.1016/0004-3702(90)90060-D.

[18] James D. Park and Adnan Darwiche. Complexity results and approximation strategies for map explanations. *J. Artif. Int. Res.*, 21(1):101133, 2004. ISSN 1076-9757.

[19] Cassio P. de Campos. New complexity results for MAP in bayesian networks. In Toby Walsh, editor, *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 2100–2106. IJCAI/AAAI, 2011. doi: 10.5591/978-1-57735-516-8/IJCAI11-351. URL https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-351.

[20] Diarmaid Conaty, Cassio P. de Campos, and Denis Deratani Mauá. Approximation complexity of maximum A posteriori inference in sum-product networks. In Gal Elidan, Kristian Kersting, and Alexander T. Ihler, editors, *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017, Sydney, Australia, August 11-15, 2017*. AUAI Press, 2017. URL http://auai.org/uai2017/proceedings/papers/109.pdf.

[21] Robert Peharz. *Foundations of Sum-Product Networks for Probabilistic Modeling*. PhD thesis, Medical University of Graz, 2015.

[22] Rina Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113(1–2):41–85, 1999. doi: 10.1016/S0004-3702(99)00061-X.

[23] 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/.

[24] 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. AUAI Press, 2012. URL https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=2327&proceeding_id=28.

[25] Andrea Montanari, Federico Ricci-Tersenghi, and Guilhem Semerjian. Solving constraint satisfaction problems through belief propagation-guided decimation. *ArXiv preprint*, abs/0709.1667, 2007. URL https://arxiv.org/abs/0709.1667.

[26] Shaowei Cai, Chuan Luo, and Haochen Zhang. From decimation to local search and back: A new approach to maxsat. In Carles Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 571–577. ijcai.org, 2017. doi: 10.24963/ijcai.2017/80. URL https://doi.org/10.24963/ijcai.2017/80.

[27] Rina Dechter. Mini-buckets: a general scheme for generating approximations in automated reasoning. In *Proceedings of the Fifteenth International Joint Conference on Artifical Intelligence - Volume 2*, IJCAI'97, page 12971302, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc. ISBN 15558604804.

[28] Sajjad Siddiqi and Jinbo Huang. Variable and value ordering for mpe search. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence (UAI)*, Arlington, Virginia, USA, 2008. AUAI Press.

[29] Radu Marinescu and Rina Dechter. Dynamic orderings for AND/OR branch-and-bound search in graphical models. In *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI)*, 2006.

[30] Radu Marinescu, Kalev Kask, and Rina Dechter. Systematic versus nonsystematic search algorithms for most probable explanations. In *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2003.

[31] Elias Boutros Khalil, Pierre Le Bodic, Le Song, George L. Nemhauser, and Bistra Dilkina. Learning to branch in mixed integer programming. In Dale Schuurmans and Michael P. Wellman, editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 724–731. AAAI Press, 2016. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12514.

[32] 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.

[33] Yuan Sun, Xiaodong Li, and Andreas Ernst. Using statistical measures and machine learning for graph reduction to solve maximum weight clique problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(5):1746–1760, 2021. doi: 10.1109/TPAMI.2019.2954827.

[34] 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.

[35] Shivvrat Arya, Tahrima Rahman, and Vibhav Gogate. Neural network approximators for marginal map in probabilistic circuits. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'24. AAAI Press, 2024. ISBN 978-1-57735-887-9. doi: 10.1609/aaai.v38i10.28966. URL https://doi.org/10.1609/aaai.v38i10.28966.

[36] Shivvrat Arya, Tahrima Rahman, and Vibhav Gogate. A neural network approach for efficiently answering most probable explanation queries in probabilistic models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 33538–33601. Curran Associates, Inc., 2024.

[37] Shivvrat Arya, Tahrima Rahman, and Vibhav Giridhar Gogate. SINE: Scalable MPE inference for probabilistic graphical models using advanced neural embeddings. In *The 28th International Conference on Artificial Intelligence and Statistics*, 2025.

[38] Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations*, 2013.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[40] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.90. URL https://doi.org/10.1109/CVPR.2016.90.

[41] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. ISSN 1533-7928.

[42] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, page 129136, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595937933. doi: 10.1145/1273496.1273513. URL https://doi.org/10.1145/1273496.1273513.

[43] Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997. doi: 10.1023/A:1007379606734.

[44] G. Elidan and A. Globerson. *The 2010 UAI Approximate Inference Challenge*. 2010.

[45] Rina Dechter, Alexander Ihler, Vibhav Gogate, Junkyu Lee, Bobak Pezeshki, Annie Raichev, and Nick Cohen. UAI 2022 competition, 2022. URL https://uaicompetition.github.io/uci-2022/.

[46] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

[47] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, 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 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 8024–8035, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

[48] Kalev Kask and Rina Dechter. Stochastic local search for bayesian network. In David Heckerman and Joe Whittaker, editors, *Proceedings of the Seventh International Workshop on Artificial Intelligence and Statistics*, volume R2 of *Proceedings of Machine Learning Research*. PMLR, 03–06 Jan 1999. URL https://proceedings.mlr.press/r2/kask99a.html. Reissued by PMLR on 20 August 2020.

[49] Tobias Achterberg, Timo Berthold, and Gregor Hendel. *Rounding and Propagation Heuristics for Mixed Integer Programming*, page 7176. 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.

[50] 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 309311, 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.

[51] Chris Wallace. Zi round, a mip rounding heuristic. *Journal of Heuristics*, 16(5):715722, 2009. ISSN 1572-9397. doi: 10.1007/s10732-009-9114-6. URL http://dx.doi.org/10.1007/s10732-009-9114-6.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: [NA]

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: [NA]

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: [NA]

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: [NA]

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We will release the code as part of our supplementary materials. Furthermore, the UAI benchmark models used in the paper for experiments are available online.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so No is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide most of the details in the main paper and additional details in the supplementary materials

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: [NA]

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: [NA]

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: [NA]

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: Our paper improves the speed, accuracy and scalability of Branch & Bound and other exact solver for MPE inference. Since algorithms already exist for solving MPE tasks, we do not perceive any negative or positive societal impacts beyond what currently exists.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: [NA]

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: [NA]

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: [NA]

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Description of the probabilistic graphical models

Table 2 summarizes the networks used in our experiments, listing the number of variables and factors in each. The experiments were conducted on high-treewidth probabilistic graphical models (PGMs) of varying sizes, with the number of variables ranging from 90 to 1440 and factor counts covering a similar range. The table also lists the network aliases used throughout the paper for ease of reference.

Table 2: Summary of networks used in our experiments, including the number of variables and factors for each.

Network Name	Network Alias	Number of Factors	Number of Variables
BN_12	BN 12	90	90
BN_9	BN 9	105	105
BN_13	BN 13	125	125
grid20x20.f5.wrap	Grid 20	1200	400
BN_65	BN 65	440	440
BN_59	BN 59	540	540
BN_53	BN 53	561	561
BN_49	BN 49	661	661
BN_61	BN 61	667	667
BN_45	BN 45	880	880
Promedas_68	Prm 68	1022	1022
Promedas_60	Prm 60	1076	1076
BN_30	BN 30	1156	1156
BN_32	BN 32	1440	1440

B Hyperparameter details

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

The neural networks in L2C-OPT and L2C-RANK use 256-dimensional embeddings, two multi-head attention layers, and 15 skip-connection blocks. Dense layers have 512 units with 0.1 dropout [41] and ReLU activations. We trained the models using Adam [46] with a learning rate of 8×10^{-4}, an exponential decay rate of 0.97, a batch size of 128, and early stopping after 5 stagnant epochs (maximum 50 epochs). All models were implemented in PyTorch [47] and executed on an NVIDIA A40 GPU. After training, we evaluated the model on 1,000 MPE queries with a query ratio 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 λ_{opt} and λ_{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 Extending L2C to multi-valued and continuous domains

While the main formulation of *Learning to Condition* (L2C) focuses on binary variables for simplicity, the framework generalizes naturally to multi-valued discrete and continuous domains with minor architectural and procedural modifications.

C.1 Multi-valued discrete variables

For multi-valued variables, we define a variable-value pair for each possible assignment. During data collection, the same procedure is used to gather solver traces, now over the expanded set of candidate pairs. The attention-based scoring architecture remains unchangedeach (variable, value) pair is embedded independently and scored through the same optimality and simplification heads. This formulation allows L2C to reason jointly over variablevalue combinations without requiring major adjustments to training or inference. At inference time, greedy or beam-based selection can be

performed over all pairs, and conditioning proceeds as in the binary case. The learned heuristic thus generalizes directly to categorical domains.

C.2 Continuous variables

Directly applying L2C to continuous domains is infeasible because the set of possible values is uncountable. To adapt, we discretize each continuous variable into a finite set of representative values (e.g., via uniform or quantile-based binning). The data collection process then conditions on these discretized values and records solver statistics as in the discrete case. During training, embeddings are created for each (variable, discretized value) pair, and the existing attention-based network and loss objectives are applied without modification. To mitigate discretization artifacts, soft-label targets or local smoothing can be used to ensure smooth transitions across adjacent bins.

At inference time, the model selects promising (variable, value) pairs based on the learned scores. After conditioning on discretized values, local continuous optimization methods (e.g., gradient descent, coordinate search) can optionally refine assignments within selected regions, improving precision beyond discretization resolution. This hybrid strategy allows L2C to extend its learned conditioning capabilities to continuous-variable inference while maintaining compatibility with existing solver frameworks.

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 [23] is used as the oracle. For each method and network, we report the average percentage gap in log-likelihood. The y-axis in Figures 5 to 18 shows the average percentage gap in log-likelihood scores, computed as:

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

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

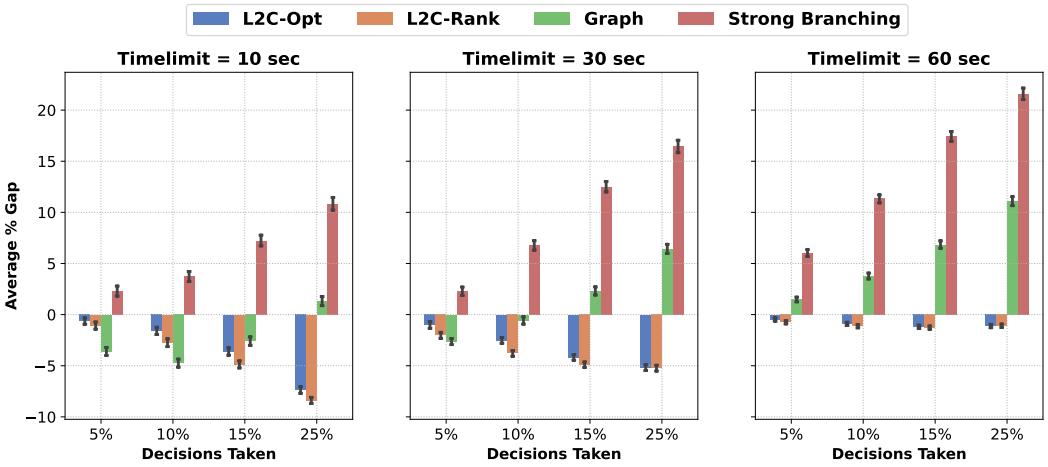


Figure 5: Average percentage gap on the BN 12 network for greedy conditioning using our methods, L2C-OPT and L2C-RANK, and baseline approaches across varying time budgets and numbers of decisions. More negative values indicate better performance.

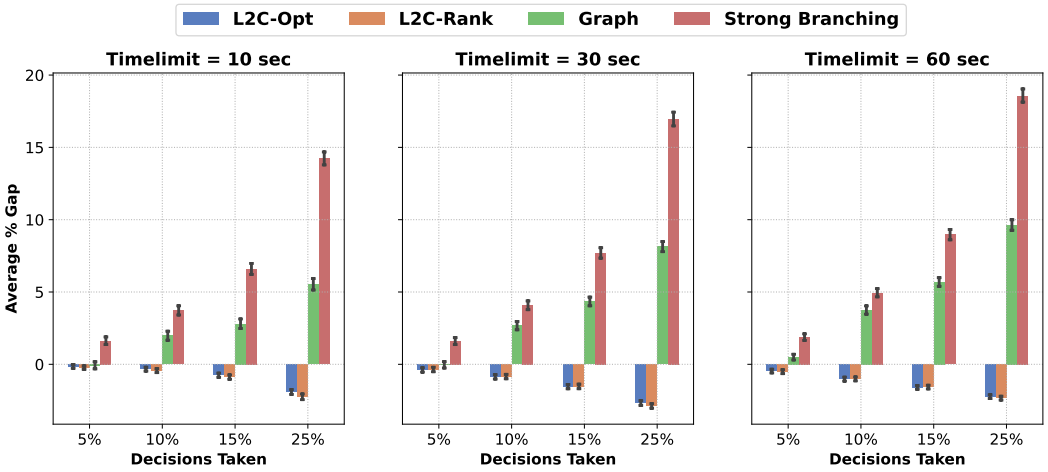


Figure 6: Average percentage gap on the BN 9 network for greedy conditioning using our methods, L2C-OPT and L2C-RANK, and baseline approaches across varying time budgets and numbers of decisions. More negative values indicate better performance.

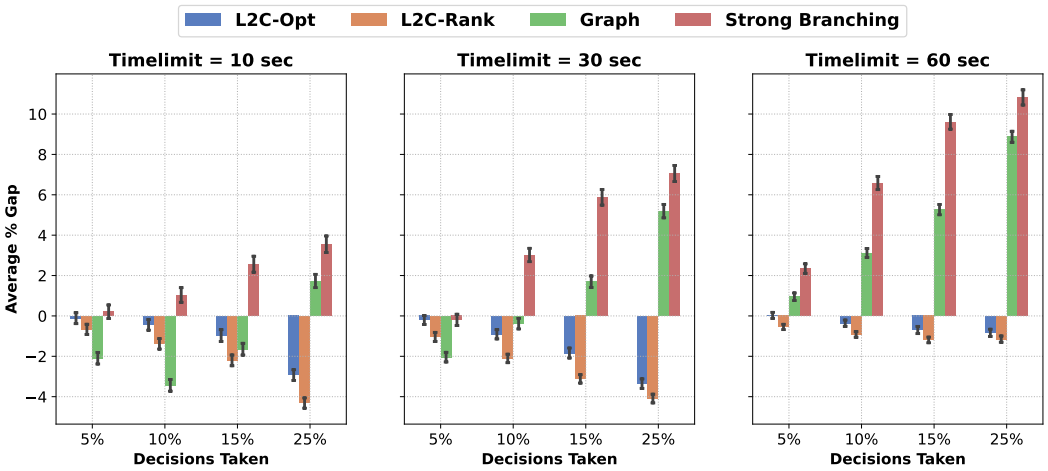


Figure 7: Average percentage gap on the BN 13 network for greedy conditioning using our methods, L2C-OPT and L2C-RANK, and baseline approaches across varying time budgets and numbers of decisions. More negative values indicate better performance.

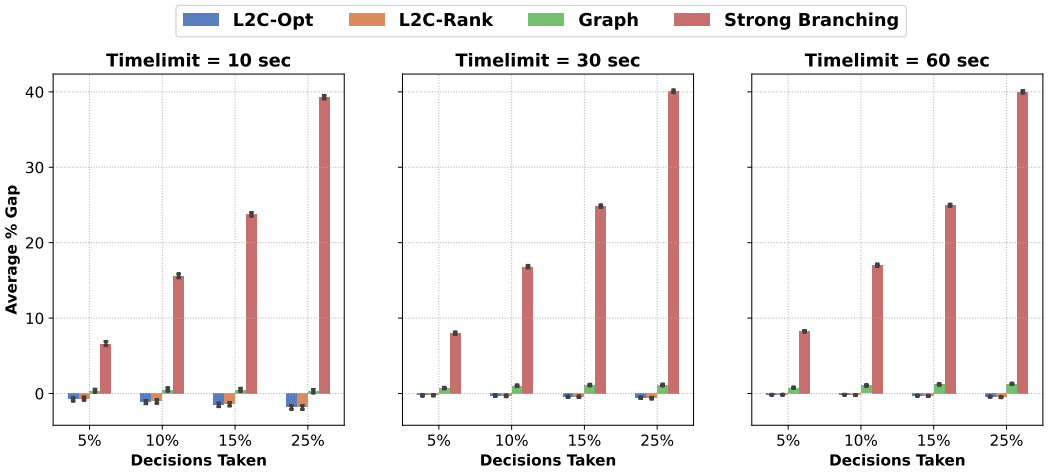


Figure 8: Average percentage gap on the Grid 20 network for greedy conditioning using our methods, L2C-OPT and L2C-RANK, and baseline approaches across varying time budgets and numbers of decisions. More negative values indicate better performance.

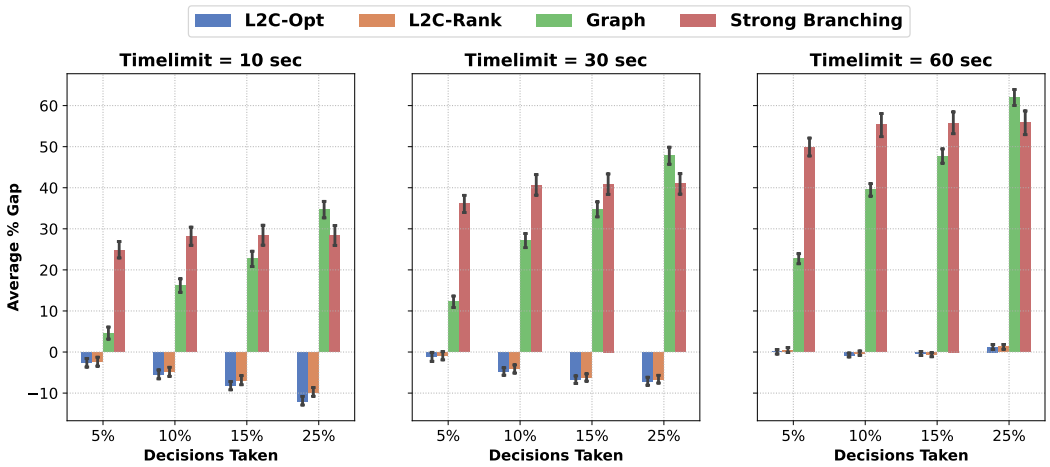


Figure 9: Average percentage gap on the BN 65 network for greedy conditioning using our methods, L2C-OPT and L2C-RANK, and baseline approaches across varying time budgets and numbers of decisions. More negative values indicate better performance.

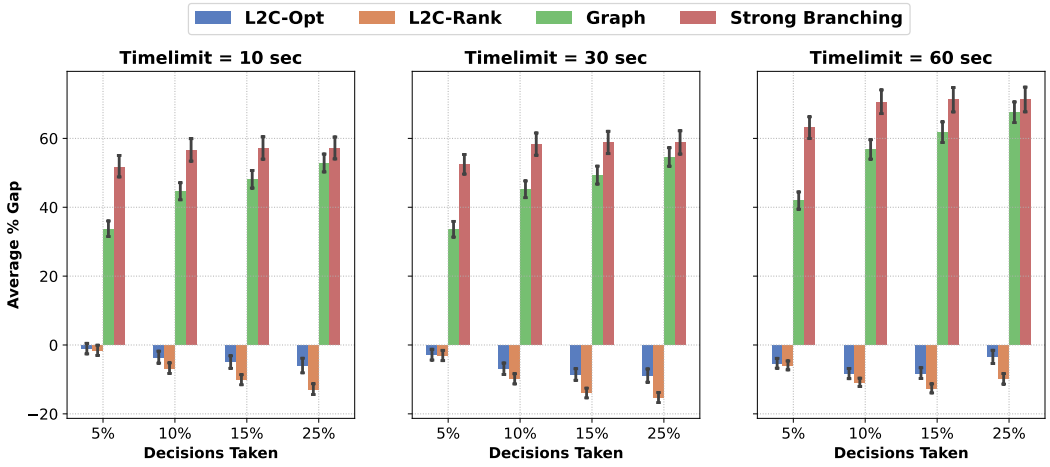


Figure 10: Average percentage gap on the BN 59 network for greedy conditioning using our methods, L2C-OPT and L2C-RANK, and baseline approaches across varying time budgets and numbers of decisions. More negative values indicate better performance.

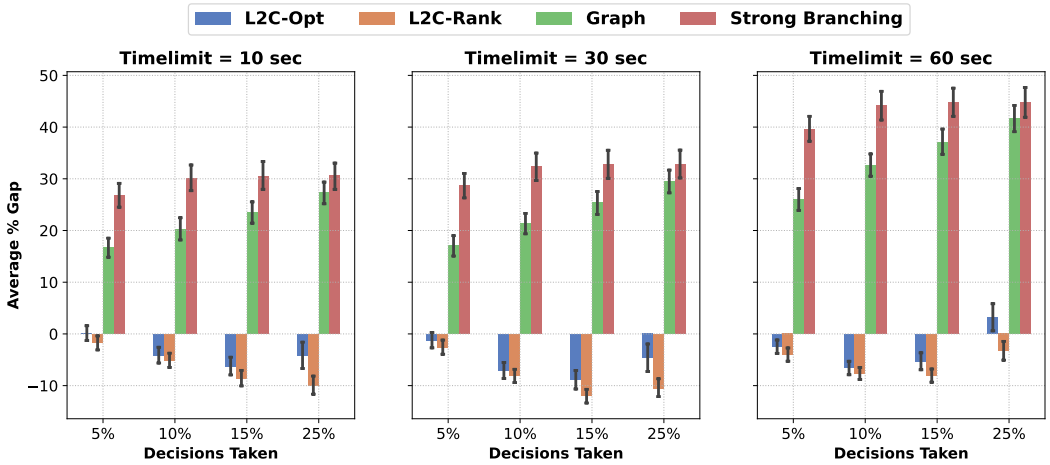


Figure 11: Average percentage gap on the BN 53 network for greedy conditioning using our methods, L2C-OPT and L2C-RANK, and baseline approaches across varying time budgets and numbers of decisions. More negative values indicate better performance.

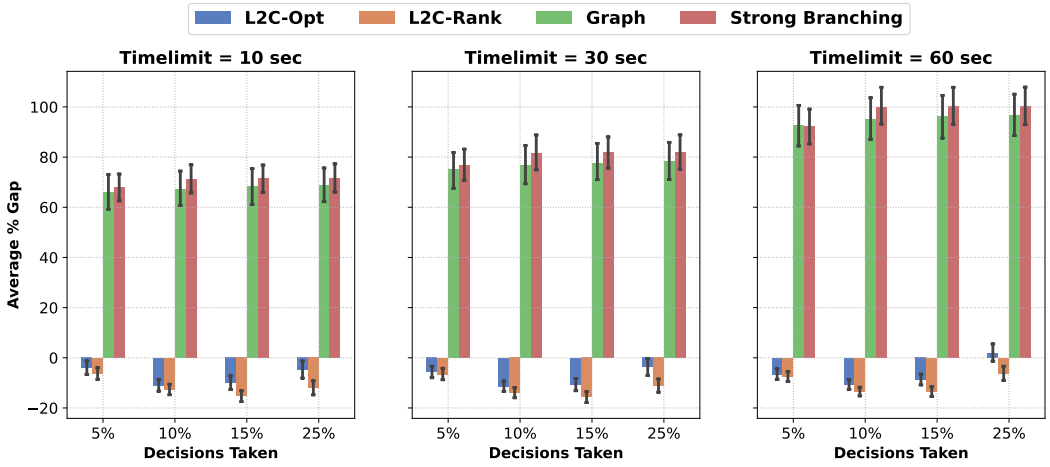


Figure 12: Average percentage gap on the BN 49 network for greedy conditioning using our methods, L2C-OPT and L2C-RANK, and baseline approaches across varying time budgets and numbers of decisions. More negative values indicate better performance.

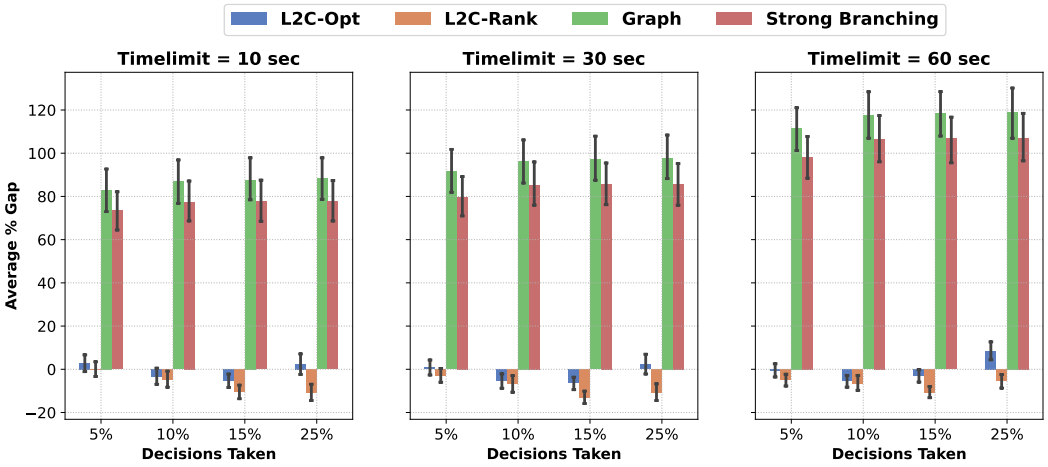


Figure 13: Average percentage gap on the BN 61 network for greedy conditioning using our methods, L2C-OPT and L2C-RANK, and baseline approaches across varying time budgets and numbers of decisions. More negative values indicate better performance.

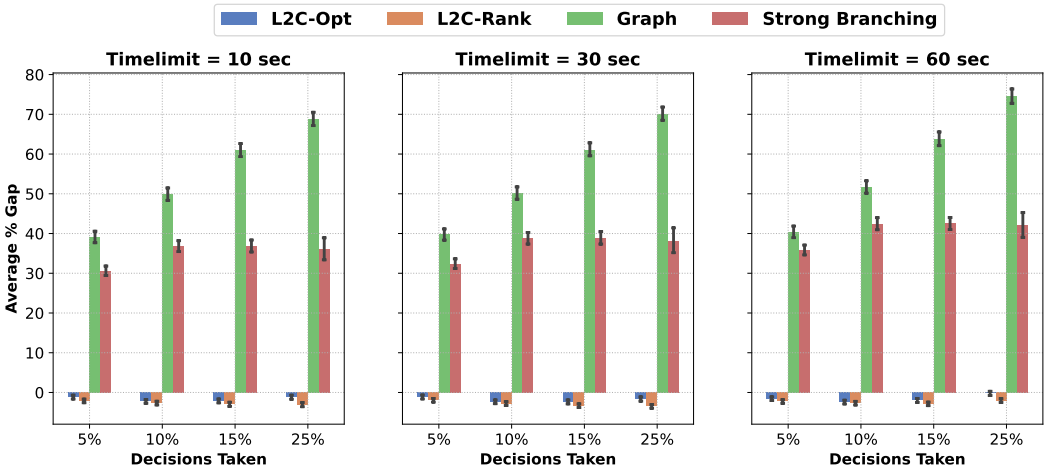


Figure 14: Average percentage gap on the BN 45 network for greedy conditioning using our methods, L2C-OPT and L2C-RANK, and baseline approaches across varying time budgets and numbers of decisions. More negative values indicate better performance.

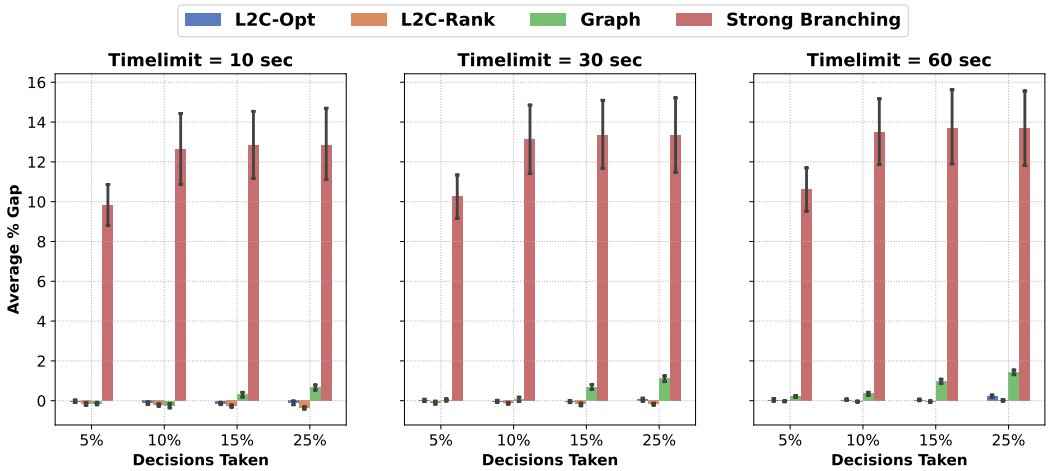


Figure 15: Average percentage gap on the Promedas 68 network for greedy conditioning using our methods, L2C-OPT and L2C-RANK, and baseline approaches across varying time budgets and numbers of decisions. More negative values indicate better performance.

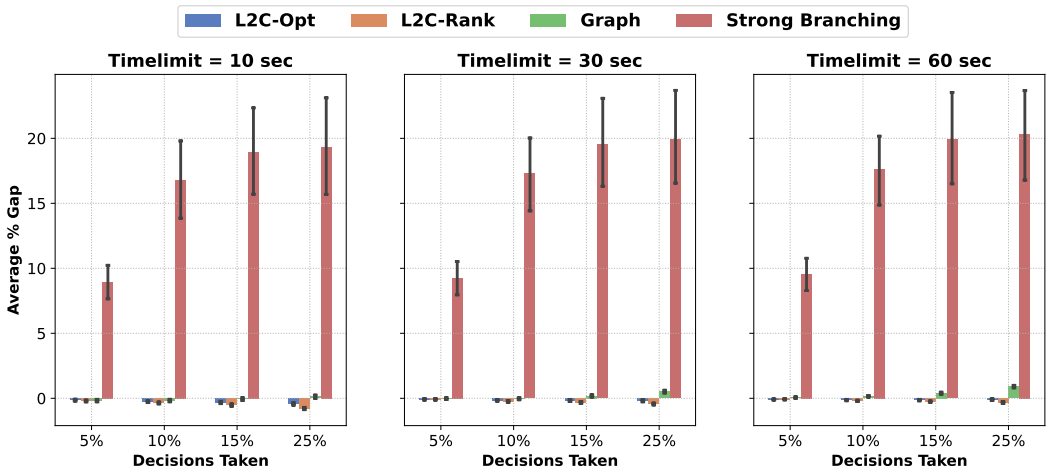


Figure 16: Average percentage gap on the Promedas 60 network for greedy conditioning using our methods, L2C-OPT and L2C-RANK, and baseline approaches across varying time budgets and numbers of decisions. More negative values indicate better performance.

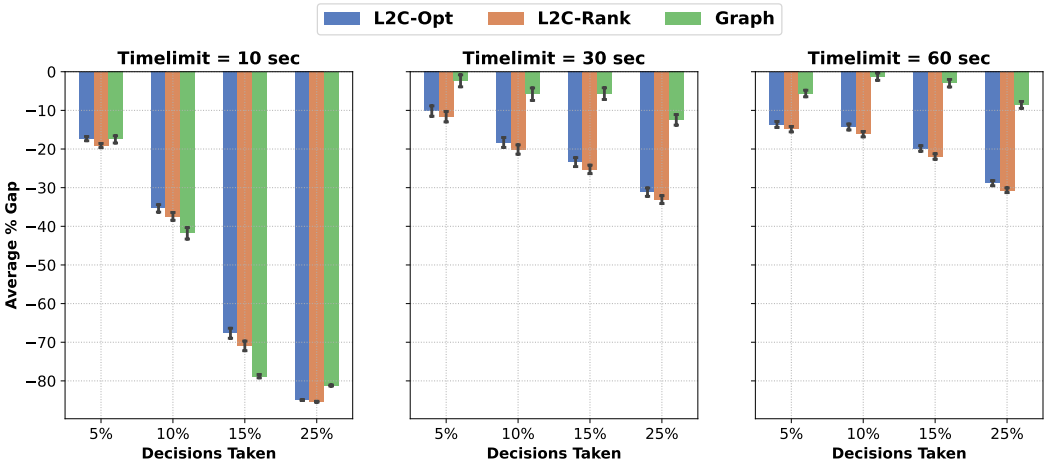


Figure 17: Average percentage gap on the BN 30 network for greedy conditioning using our methods, L2C-OPT and L2C-RANK, and baseline approaches across varying time budgets and numbers of decisions. More negative values indicate better performance.

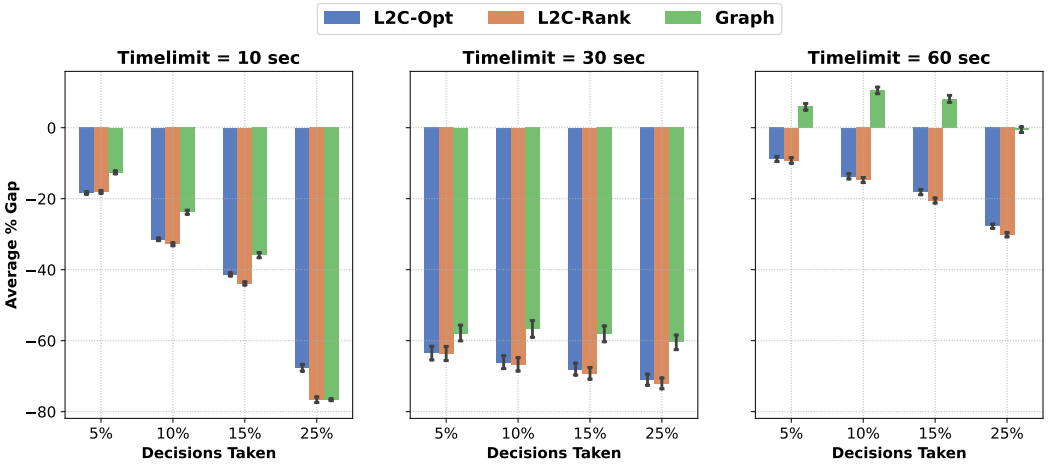


Figure 18: Average percentage gap on the BN 32 network for greedy conditioning using our methods, L2C-OPT and L2C-RANK, and baseline approaches across varying time budgets and numbers of decisions. More negative values indicate better performance.

E Conditioning performance for beam search-based conditioning

E.1 Using SCIP as oracle

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

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

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

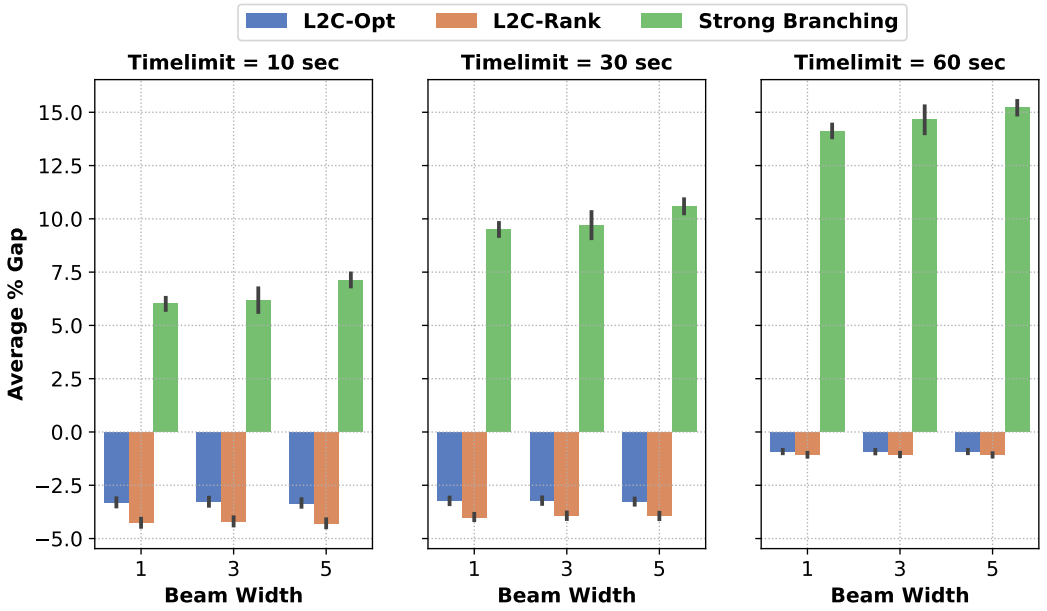


Figure 19: Average percentage gap on the BN 12 network comparing our methods to baseline approaches across varying time budgets and beam widths using beam-search based conditioning. More negative values indicate better performance.

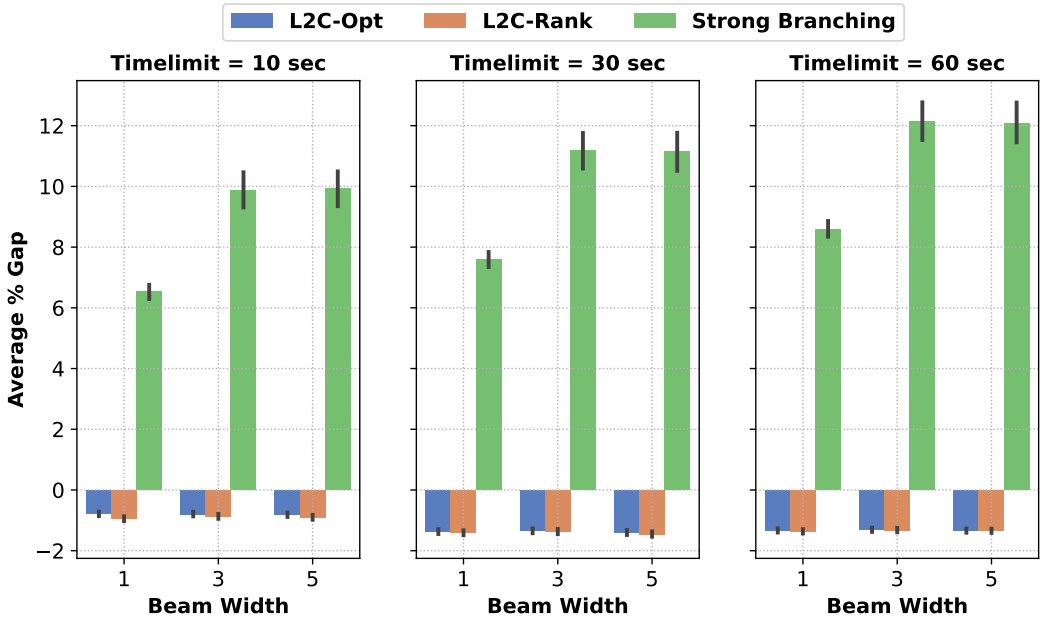


Figure 20: Average percentage gap on the BN 9 network comparing our methods to baseline approaches across varying time budgets and beam widths using beam-search based conditioning. More negative values indicate better performance.

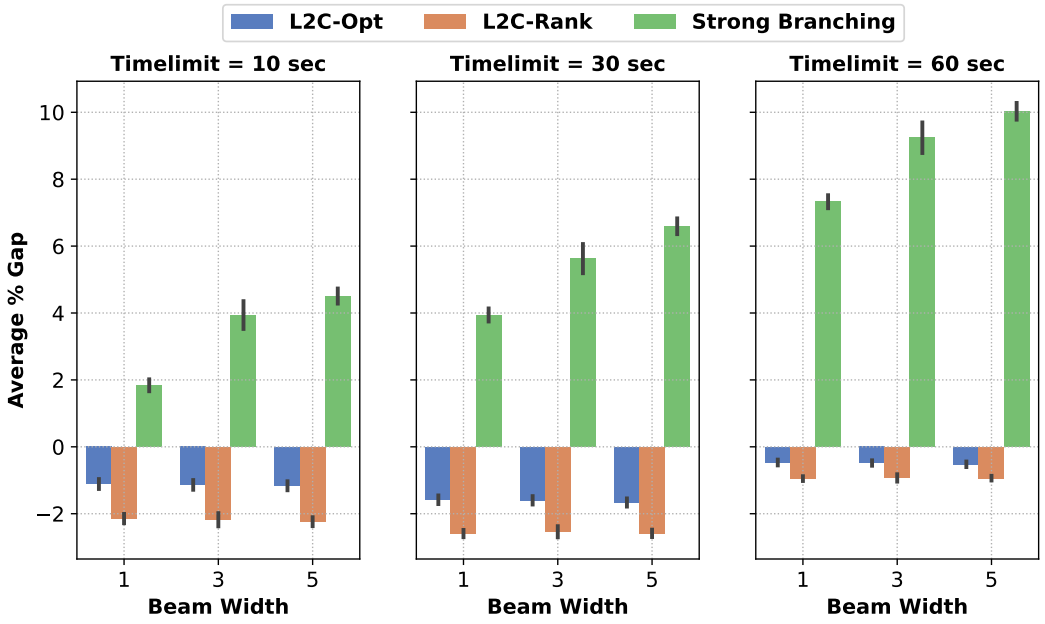


Figure 21: Average percentage gap on the BN 13 network comparing our methods to baseline approaches across varying time budgets and beam widths using beam-search based conditioning. More negative values indicate better performance.

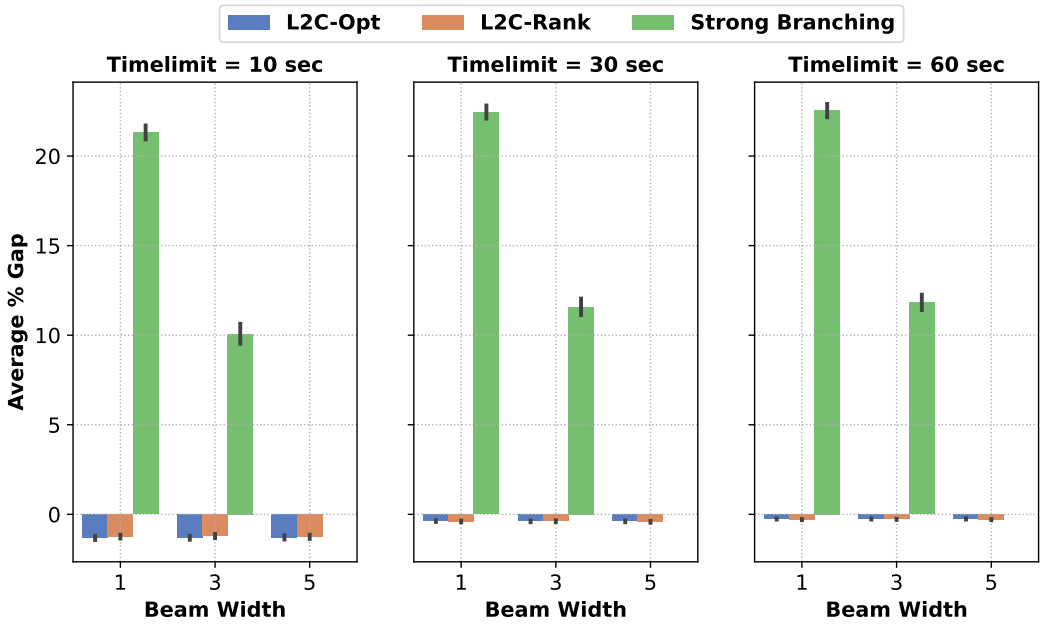


Figure 22: Average percentage gap on the Grid 20 network comparing our methods to baseline approaches across varying time budgets and beam widths using beam-search based conditioning. More negative values indicate better performance. Missing bars for strong branching indicate timeouts.

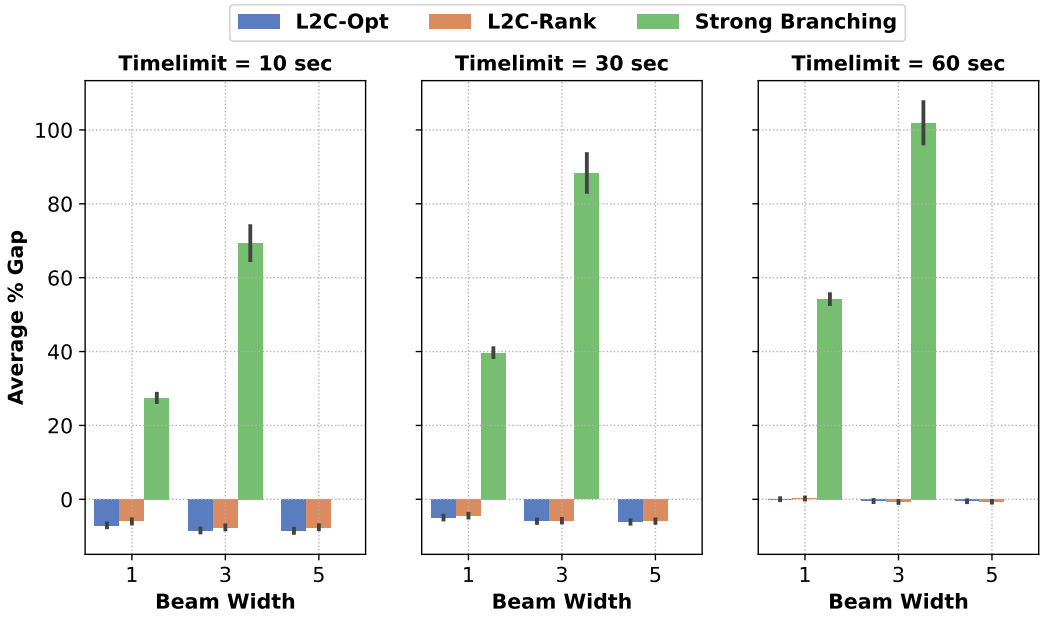


Figure 23: Average percentage gap on the BN 65 network comparing our methods to baseline approaches across varying time budgets and beam widths using beam-search based conditioning. More negative values indicate better performance. Missing bars for strong branching indicate timeouts.

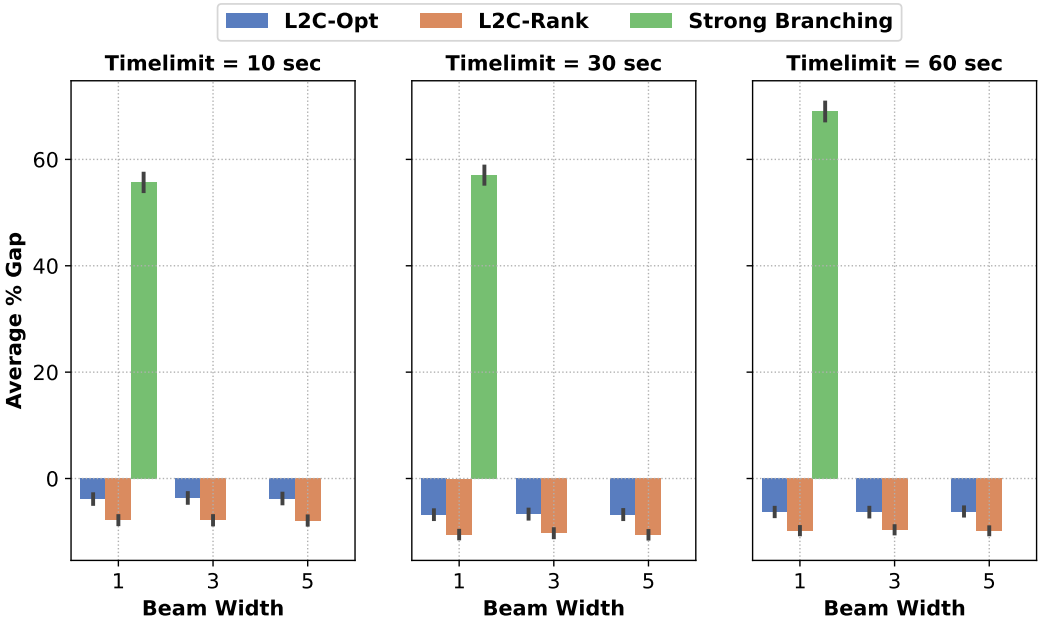


Figure 24: Average percentage gap on the BN 59 network comparing our methods to baseline approaches across varying time budgets and beam widths using beam-search based conditioning. More negative values indicate better performance. Missing bars for strong branching indicate timeouts.

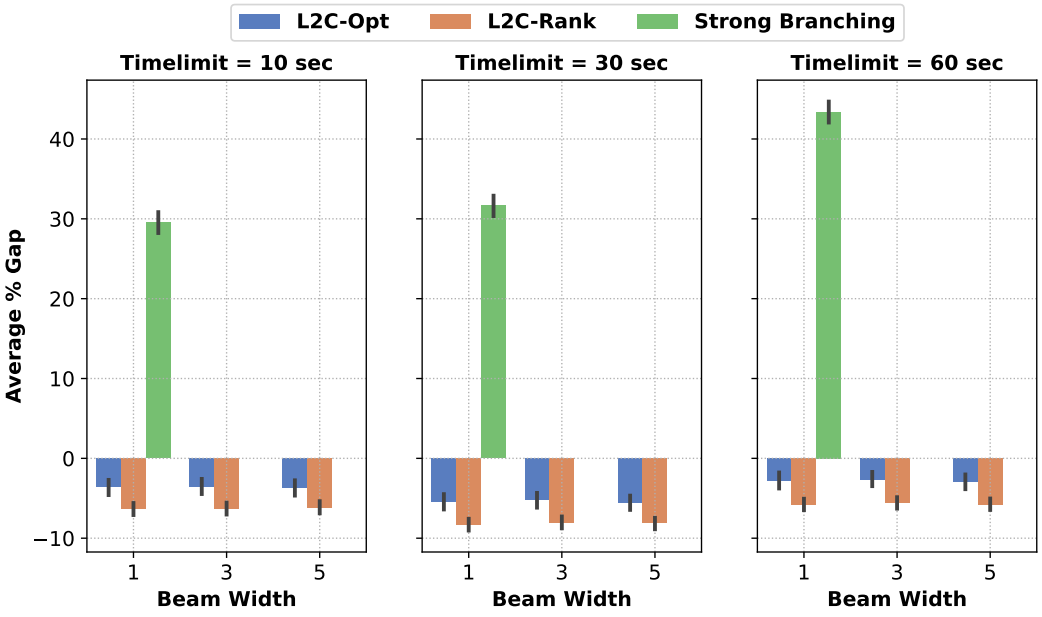


Figure 25: Average percentage gap on the BN 53 network comparing our methods to baseline approaches across varying time budgets and beam widths using beam-search based conditioning. More negative values indicate better performance. Missing bars for strong branching indicate timeouts.

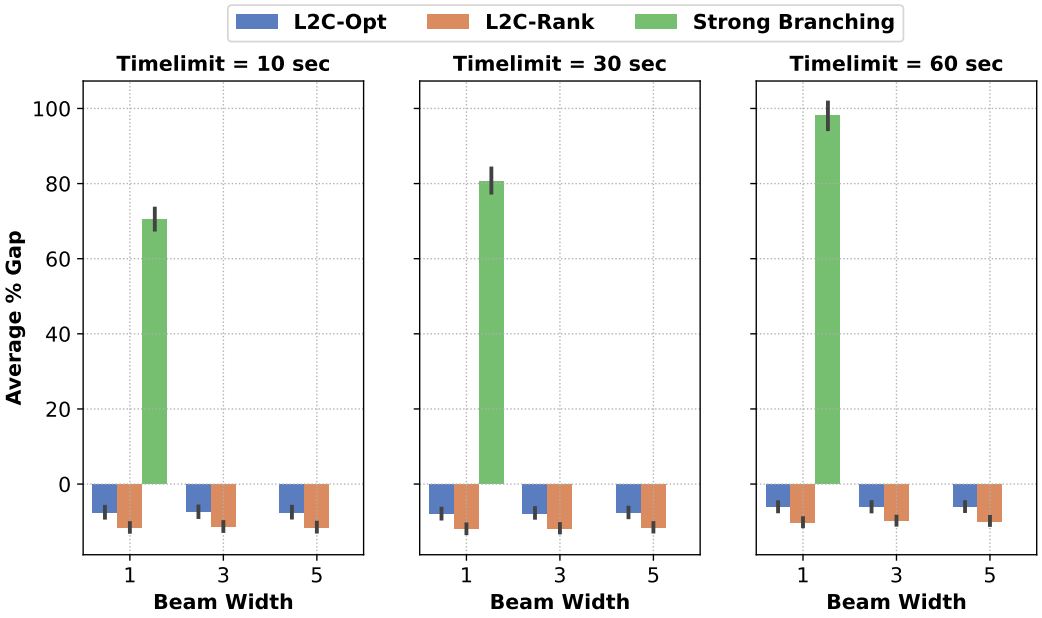


Figure 26: Average percentage gap on the BN 49 network comparing our methods to baseline approaches across varying time budgets and beam widths using beam-search based conditioning. More negative values indicate better performance. Missing bars for strong branching indicate timeouts.

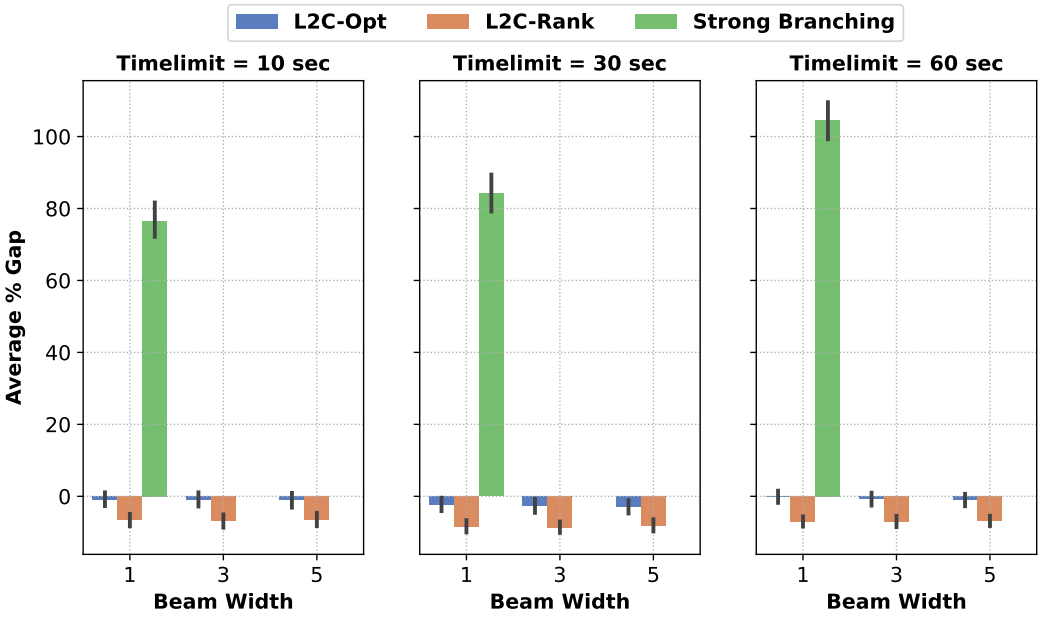


Figure 27: Average percentage gap on the BN 61 network comparing our methods to baseline approaches across varying time budgets and beam widths using beam-search based conditioning. More negative values indicate better performance. Missing bars for strong branching indicate timeouts.

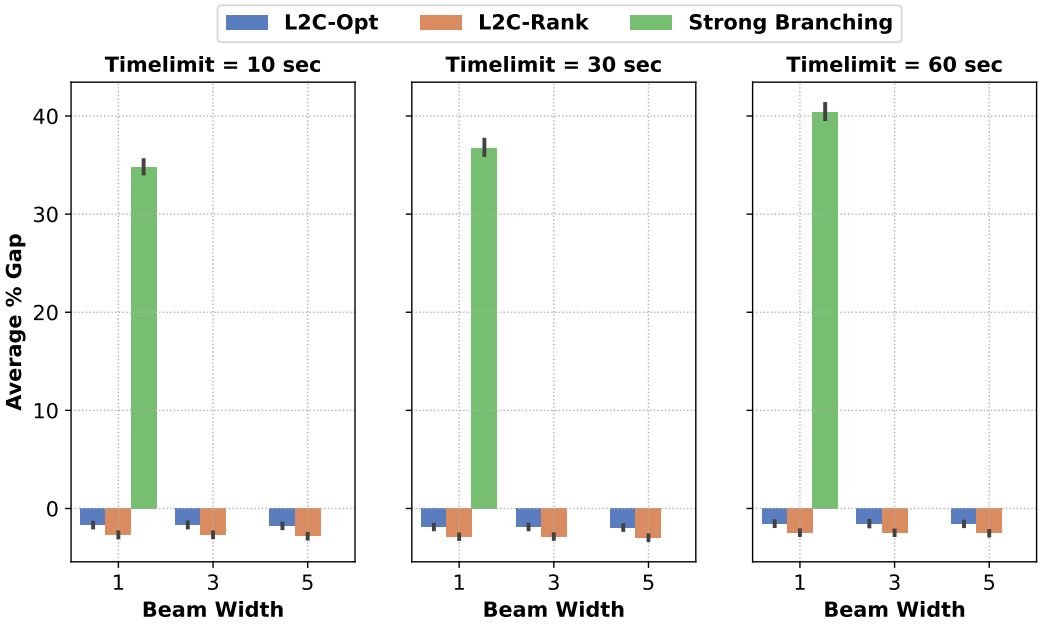


Figure 28: Average percentage gap on the BN 45 network comparing our methods to baseline approaches across varying time budgets and beam widths using beam-search based conditioning. More negative values indicate better performance. Missing bars for strong branching indicate timeouts.

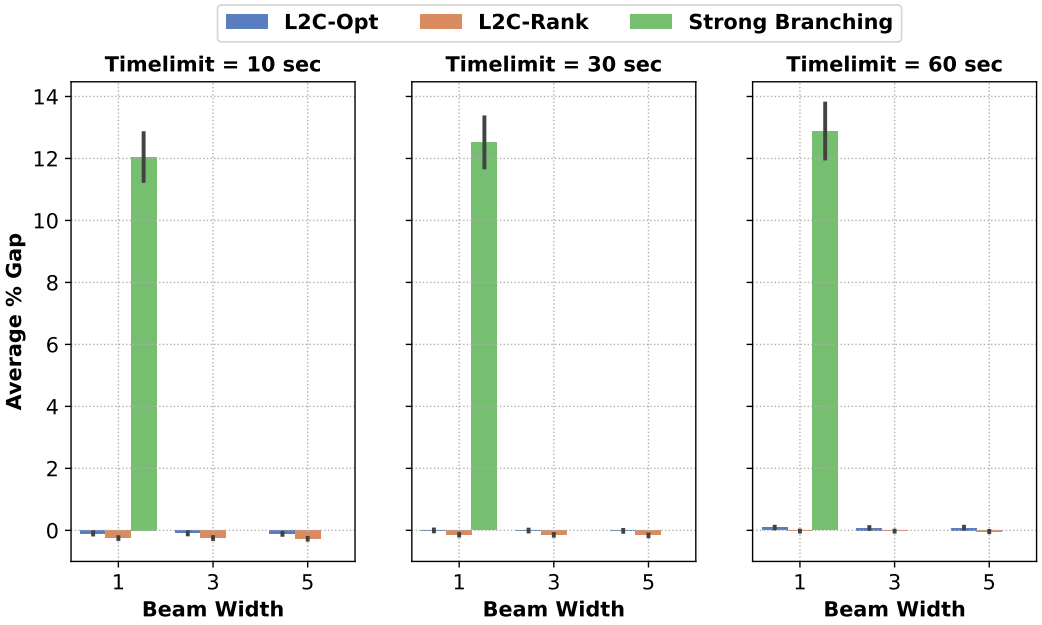


Figure 29: Average percentage gap on the Promedas 68 network comparing our methods to baseline approaches across varying time budgets and beam widths using beam-search based conditioning. More negative values indicate better performance. Missing bars for strong branching indicate timeouts.

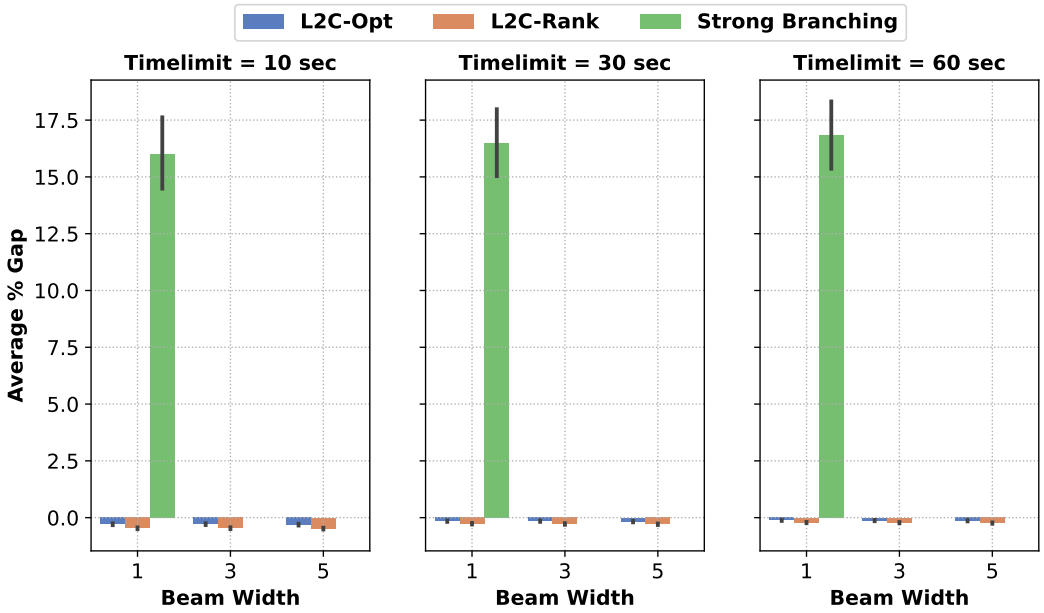


Figure 30: Average percentage gap on the Promedas 60 network comparing our methods to baseline approaches across varying time budgets and beam widths using beam-search based conditioning. More negative values indicate better performance. Missing bars for strong branching indicate timeouts.

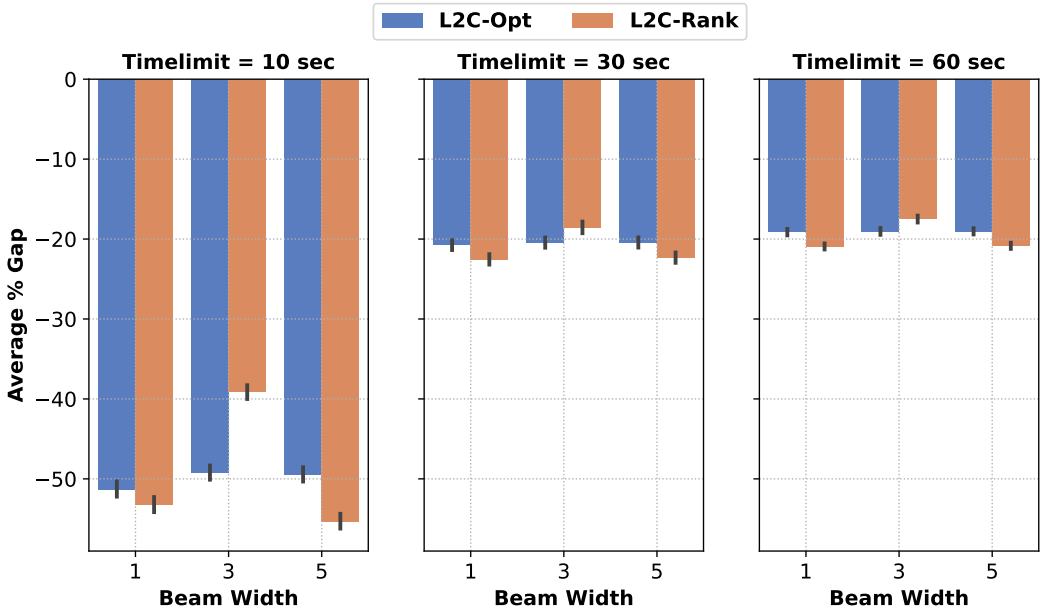


Figure 31: Average percentage gap on the BN 30 network comparing our methods to baseline approaches across varying time budgets and beam widths using beam-search based conditioning. More negative values indicate better performance. Missing bars for strong branching indicate timeouts.

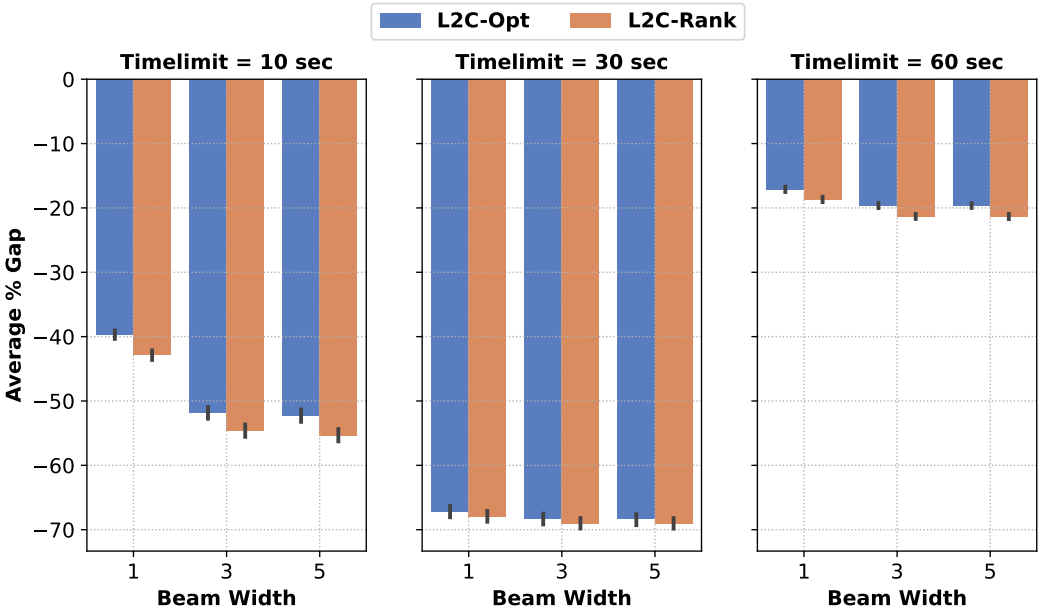


Figure 32: Average percentage gap on the BN 32 network comparing our methods to baseline approaches across varying time budgets and beam widths using beam-search based conditioning. More negative values indicate better performance. Missing bars for strong branching indicate timeouts.

E.2 Using AOBB as oracle

In this section, we evaluate the performance of beam search-based conditioning with AOBB (per Otten and Dechter [24] and implemented by Otten [3]) as the oracle, using beam widths of 3 and 5, as shown in Figures 33 and 34, respectively. Each figure plots the average solution gap against the percentage reduction in the number of search nodes across 1,000 MPE queries over all networks, with each point representing one network. The solution gap is computed using Equation 1, while node reduction is calculated using the same formula with log-likelihood scores replaced by node counts before and after conditioning.

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

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

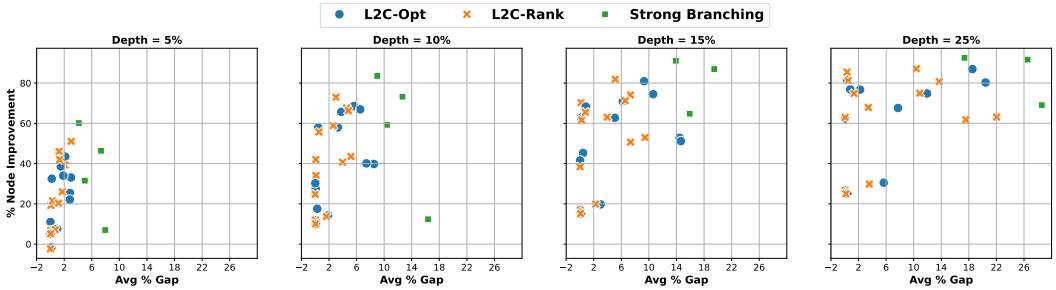


Figure 33: Beam search-based conditioning with AOBB as the oracle using a beam width of 3. Each subfigure corresponds to a fixed number of conditioning decisions. The x-axis indicates the average solution gap (lower is better), and the y-axis indicates the percentage reduction in node count (higher is better). Each point represents a single network.

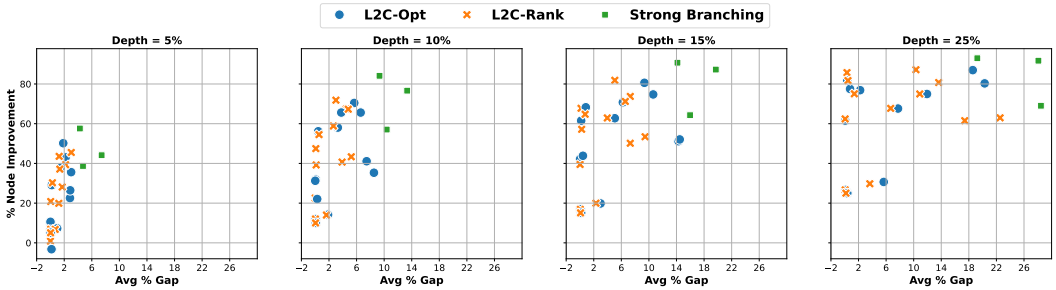


Figure 34: Beam search-based conditioning with AOBB as the oracle using a beam width of 5. Each subfigure corresponds to a fixed number of conditioning decisions. The x-axis indicates the average solution gap (lower is better), and the y-axis indicates the percentage reduction in node count (higher is better). Each point represents a single network.

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

In this section, we compare our trained neural networksused as branching rules [32] and node selection heuristics [34] within the SCIP frameworkwith SCIPs default heuristics [23, 49, 50]. We evaluate performance based on the average percentage gap in log-likelihood (LL) scores between our methods, L2C-OPT and L2C-RANK, and SCIPs default strategies on the same set of MPE queries. The gap is computed as:

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

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 SCIPs default heuristics.

As shown in Figures 35 to 48, 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 SCIPs 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.

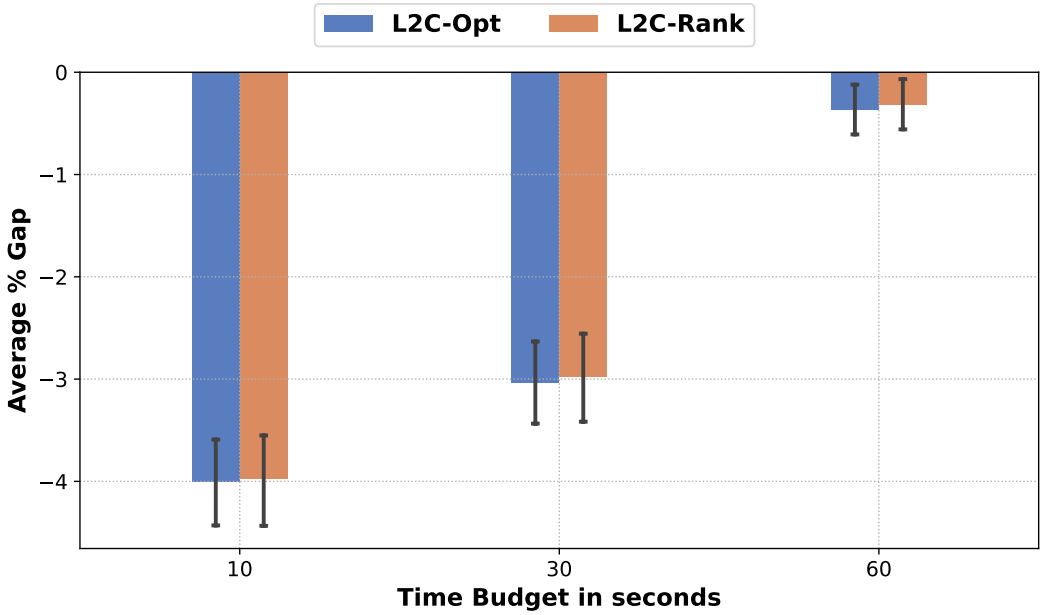


Figure 35: Comparison of SCIPs default heuristics with our neural strategies, L2C-OPT and L2C-RANK, for branching and node selection within the SCIP framework on the BN 12 network in terms of average % gap in log-likelihood. More negative values indicate better performance.

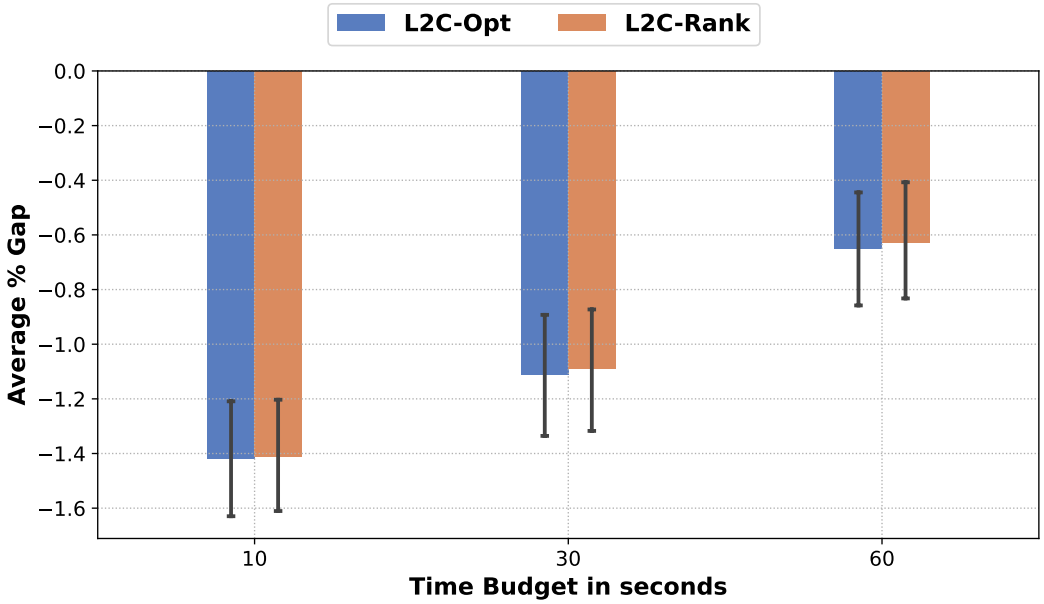


Figure 36: Comparison of SCIPs default heuristics with our neural strategies, L2C-OPT and L2C-RANK, for branching and node selection within the SCIP framework on the BN 9 network in terms of average % gap in log-likelihood. More negative values indicate better performance.

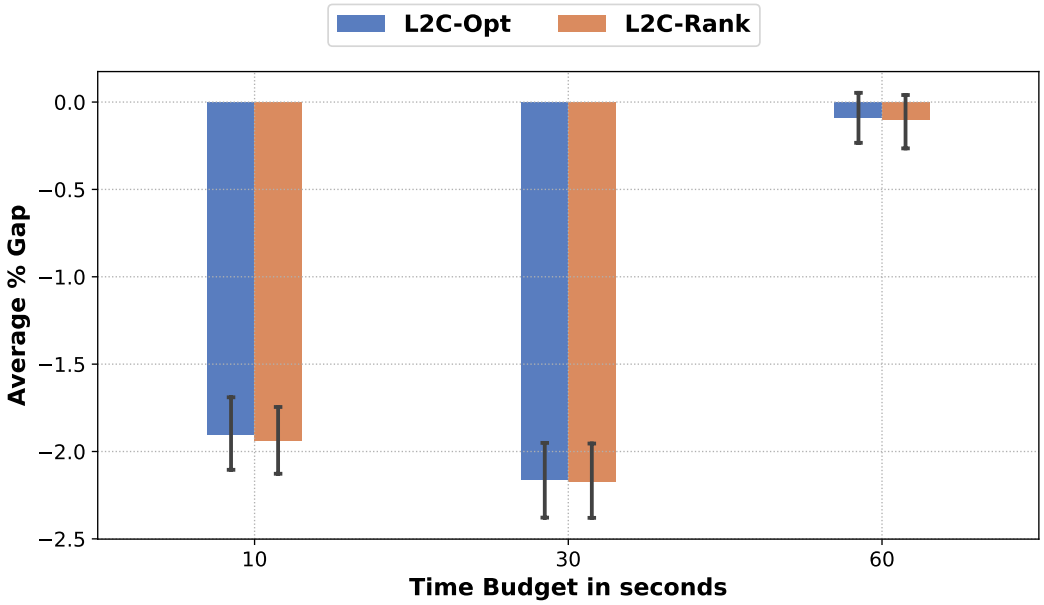


Figure 37: Comparison of SCIPs default heuristics with our neural strategies, L2C-OPT and L2C-RANK, for branching and node selection within the SCIP framework on the BN 13 network in terms of average % gap in log-likelihood. More negative values indicate better performance.

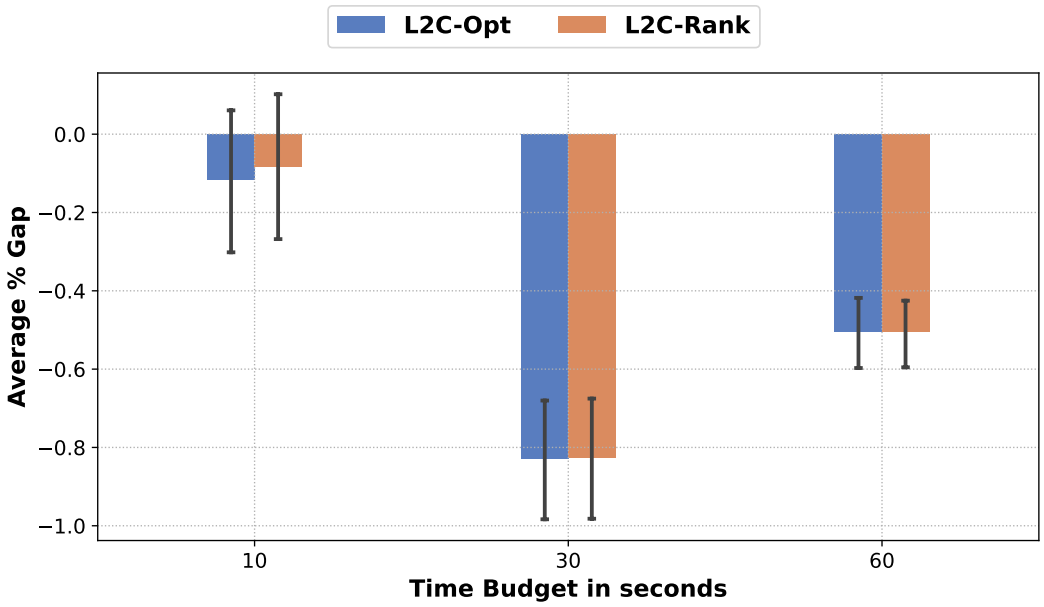


Figure 38: Comparison of SCIPs default heuristics with our neural strategies, L2C-OPT and L2C-RANK, for branching and node selection within the SCIP framework on the Grid 20 network in terms of average % gap in log-likelihood. More negative values indicate better performance.

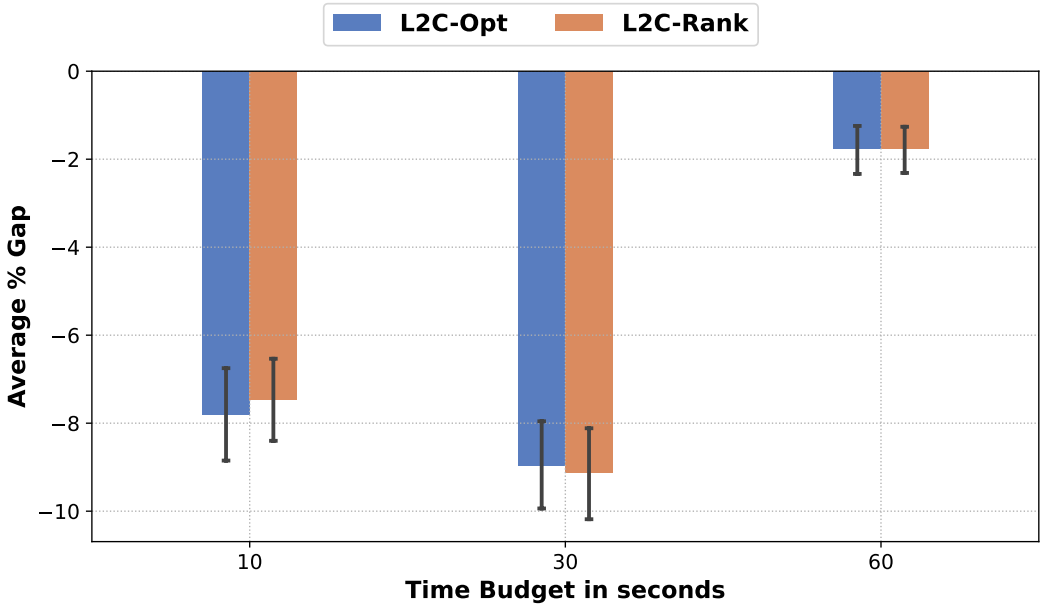


Figure 39: Comparison of SCIPs default heuristics with our neural strategies, L2C-OPT and L2C-RANK, for branching and node selection within the SCIP framework on the BN 65 network in terms of average % gap in log-likelihood. More negative values indicate better performance.

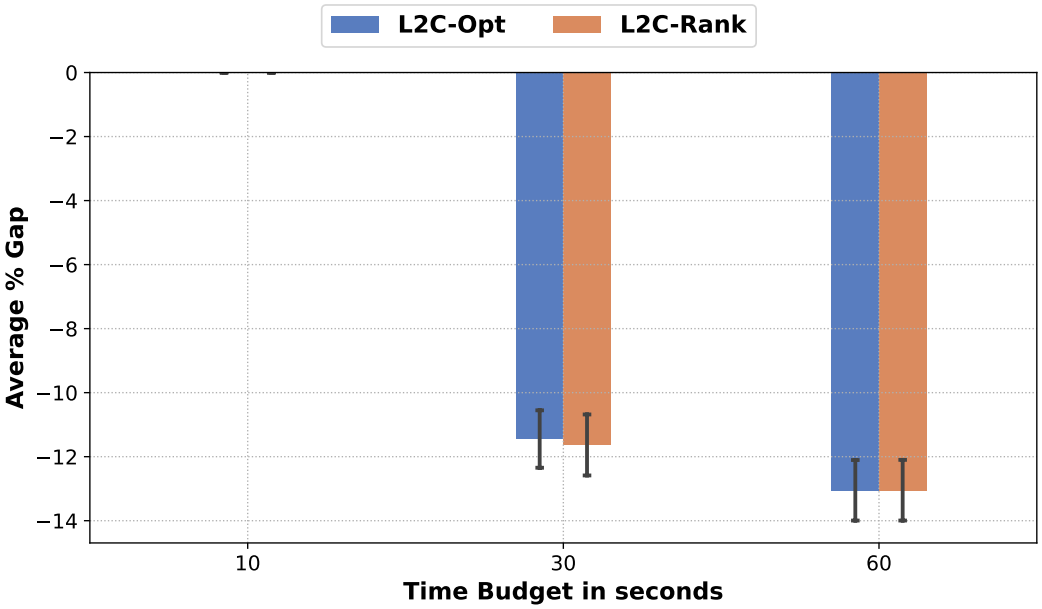


Figure 40: Comparison of SCIPs default heuristics with our neural strategies, L2C-OPT and L2C-RANK, for branching and node selection within the SCIP framework on the BN 59 network in terms of average % gap in log-likelihood. More negative values indicate better performance.

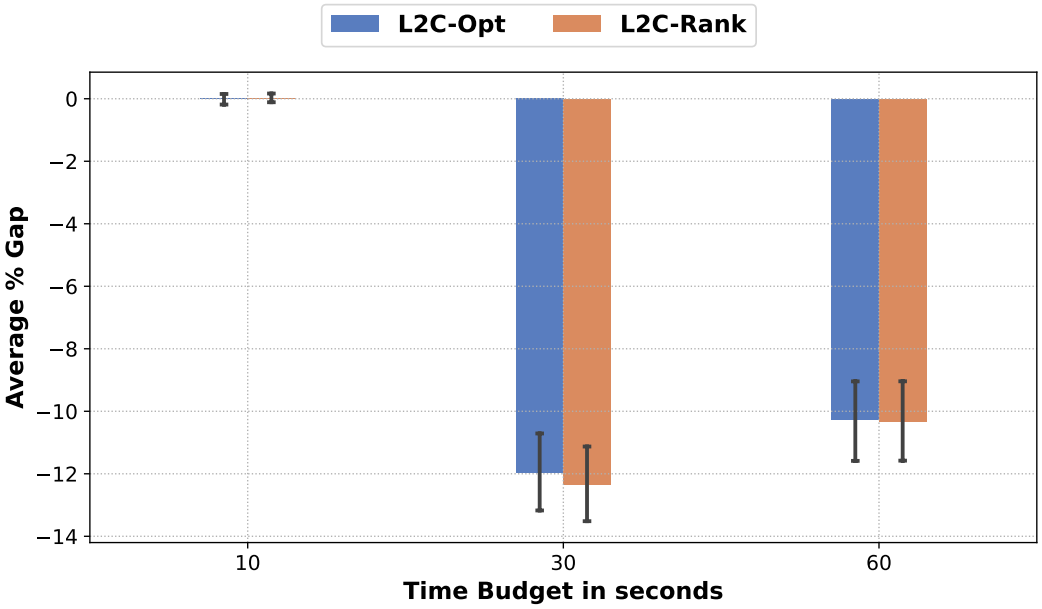


Figure 41: Comparison of SCIPs default heuristics with our neural strategies, L2C-OPT and L2C-RANK, for branching and node selection within the SCIP framework on the BN 53 network in terms of average % gap in log-likelihood. More negative values indicate better performance.

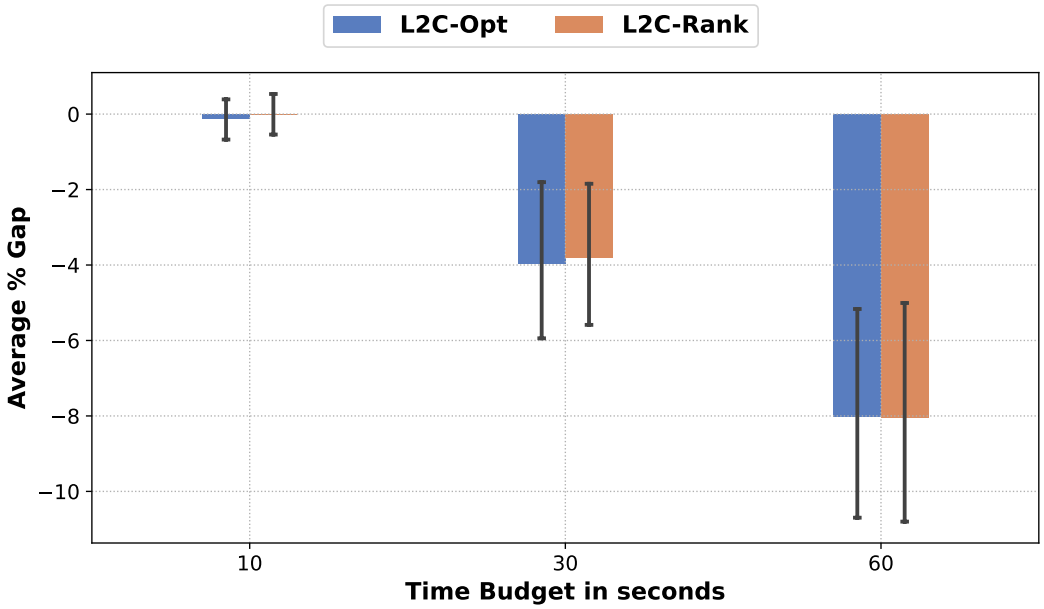


Figure 42: Comparison of SCIPs default heuristics with our neural strategies, L2C-OPT and L2C-RANK, for branching and node selection within the SCIP framework on the BN 49 network in terms of average % gap in log-likelihood. More negative values indicate better performance.

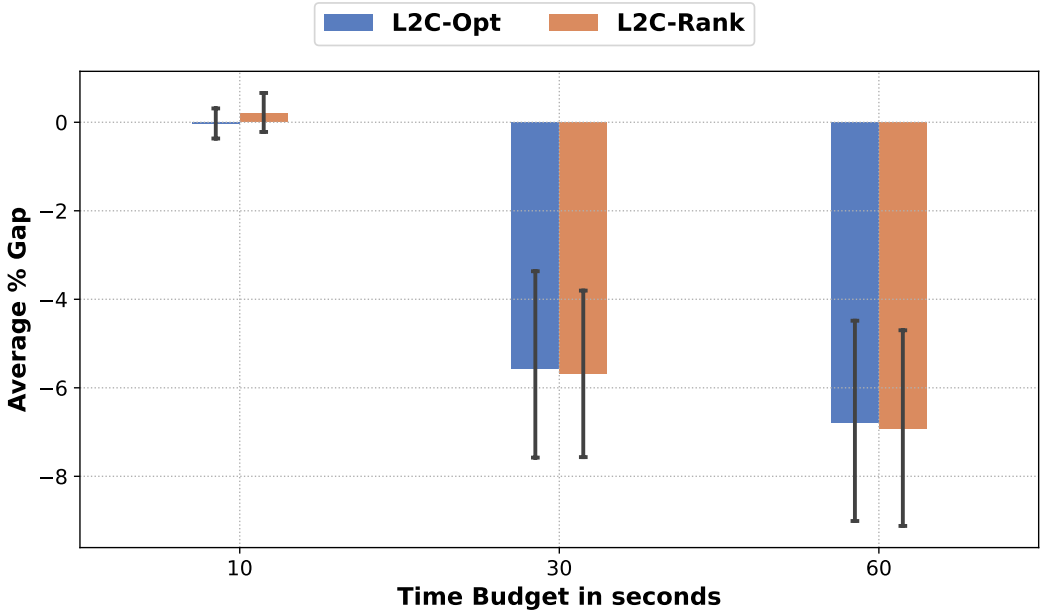


Figure 43: Comparison of SCIPs default heuristics with our neural strategies, L2C-OPT and L2C-RANK, for branching and node selection within the SCIP framework on the BN 61 network in terms of average % gap in log-likelihood. More negative values indicate better performance.

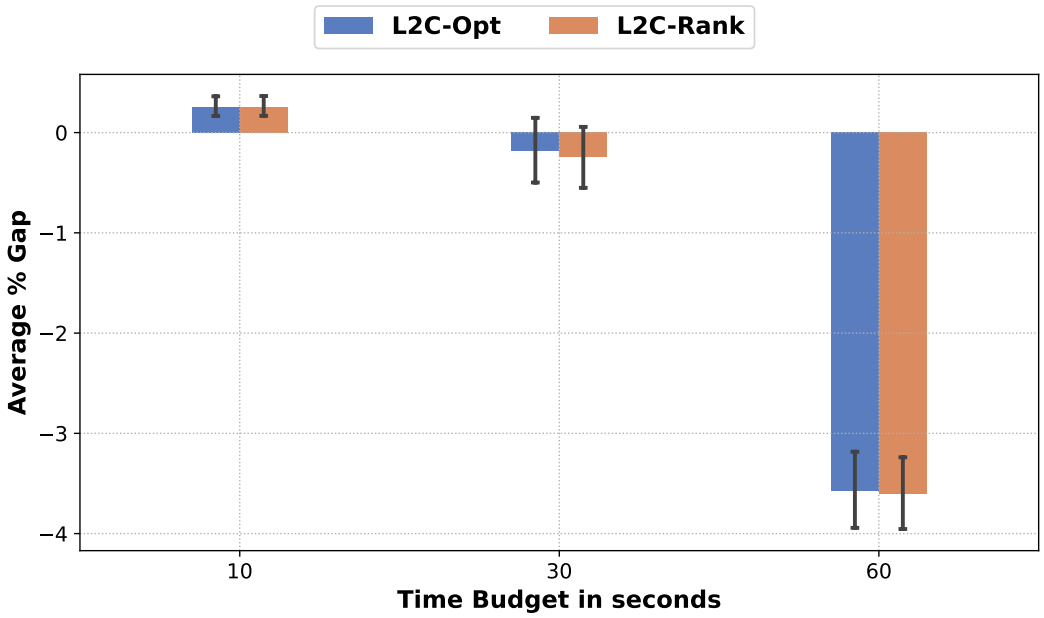


Figure 44: Comparison of SCIPs default heuristics with our neural strategies, L2C-OPT and L2C-RANK, for branching and node selection within the SCIP framework on the BN 45 network in terms of average % gap in log-likelihood. More negative values indicate better performance.

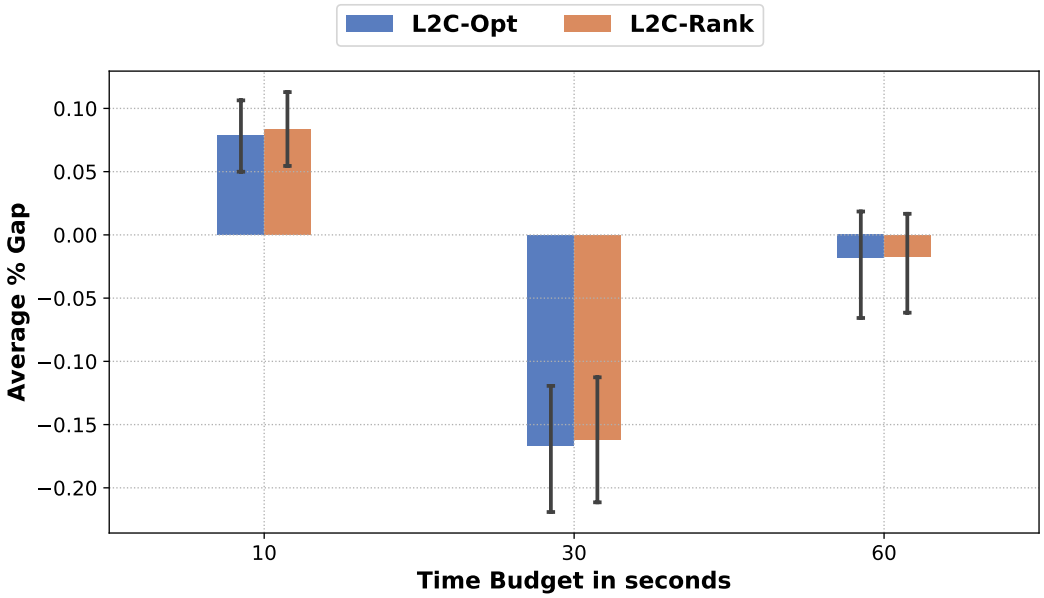


Figure 45: Comparison of SCIPs default heuristics with our neural strategies, L2C-OPT and L2C-RANK, for branching and node selection within the SCIP framework on the Promedas 68 network in terms of average % gap in log-likelihood. More negative values indicate better performance.

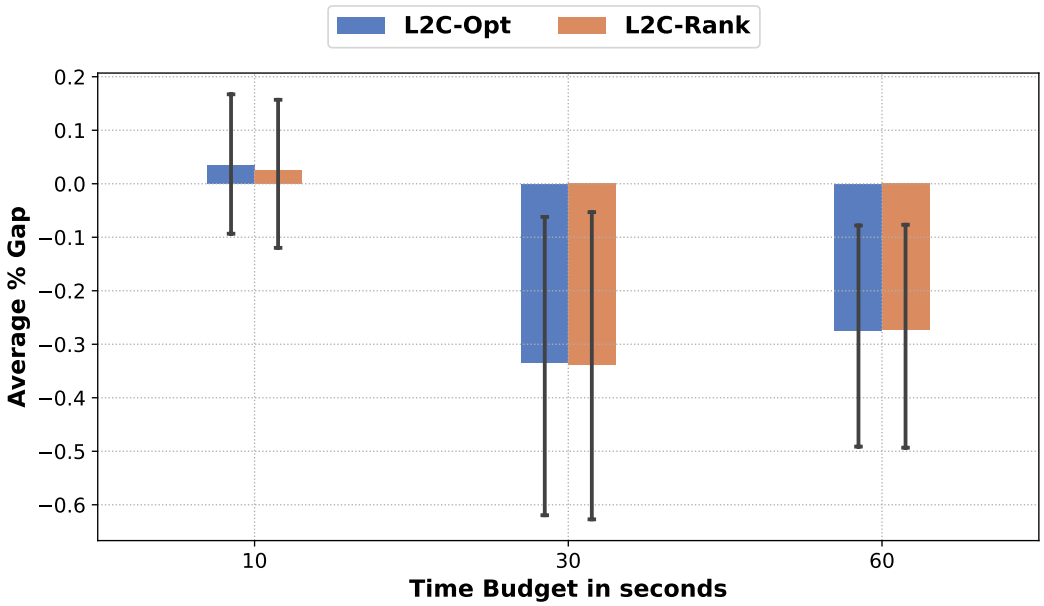


Figure 46: Comparison of SCIPs default heuristics with our neural strategies, L2C-OPT and L2C-RANK, for branching and node selection within the SCIP framework on the Promedas 60 network in terms of average % gap in log-likelihood. More negative values indicate better performance.

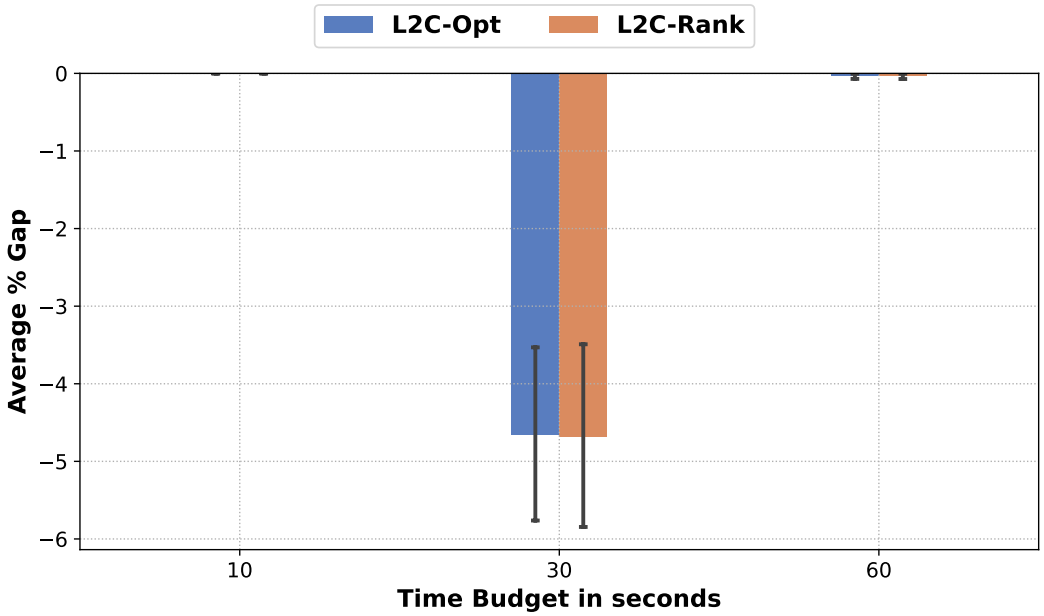


Figure 47: Comparison of SCIPs default heuristics with our neural strategies, L2C-OPT and L2C-RANK, for branching and node selection within the SCIP framework on the BN 30 network in terms of average % gap in log-likelihood. More negative values indicate better performance.

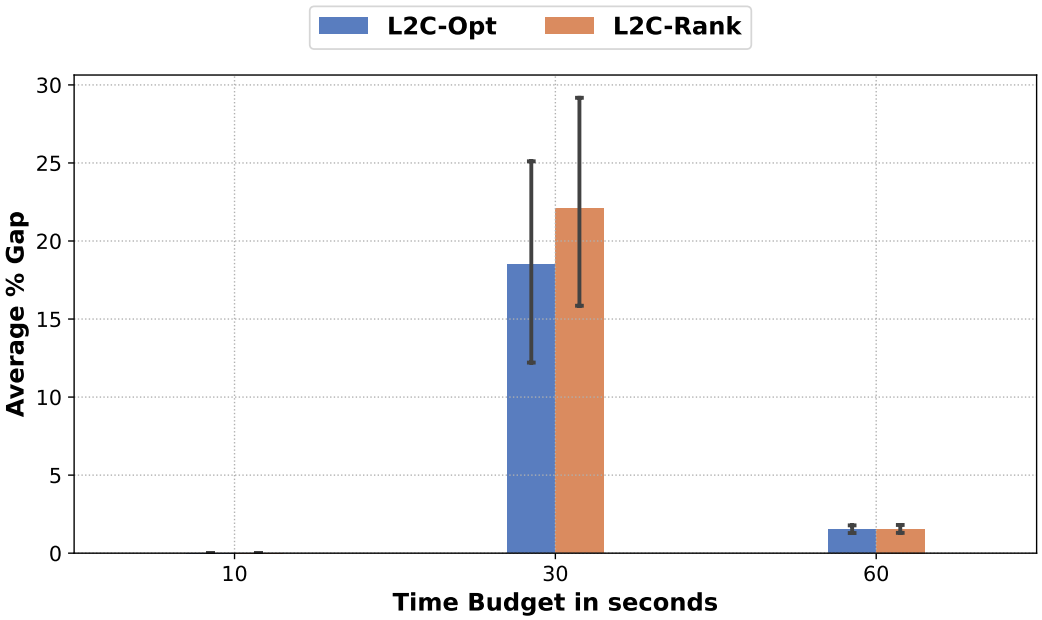


Figure 48: Comparison of SCIPs default heuristics with our neural strategies, L2C-OPT and L2C-RANK, for branching and node selection within the SCIP framework on the BN 32 network in terms of average % gap in log-likelihood. More negative values indicate better performance.