A Comprehensive Evaluation of Contemporary Machine-Learning-Based Solvers for Combinatorial Optimization

Shengyu Feng [*1] **Weiwei Sun** [*1] **Shanda Li** [1] **Ameet Talwalkar** [1] **Yiming Yang** [1]

Abstract

Machine learning (ML) has demonstrated considerable potential in supporting model design and optimization for combinatorial optimization (CO) problems. However, much of the progress to date has been evaluated on small-scale, synthetic datasets, raising concerns about the practical effectiveness of ML-based solvers in real-world, large-scale CO scenarios. Additionally, many existing CO benchmarks lack sufficient training data, limiting their utility for evaluating data-driven approaches. To address these limitations, we introduce **FrontierCO**, a comprehensive benchmark that covers eight canonical CO problem types and evaluates 16 representative ML-based solvers—including graph neural networks and large language model (LLM) agents. FRONTIERCO features challenging instances drawn from industrial applications and frontier CO research, offering both realistic problem difficulty and abundant training data. Our empirical results provide critical insights into the strengths and limitations of current ML methods, helping to guide more robust and practically relevant advances at the intersection of ML and CO.

1. Introduction

Combinatorial Optimization (CO) is a well-established research area in computer science, discrete math and operations research. It consists of finding the optimal solution in a discrete space, with various real-world applications such as allocation, routing, and planning (Korte & Vygen, 2012). CO problems are usually intractable or NP-hard (Arora & Barak, 2009), requiring overwhelming human intelligence in algorithm design. Therefore, automating the algorithm design and optimization for CO problems has become increasingly important and received widespread attention from the machine learning (ML) community.

Contemporary ML-based solvers for CO can be broadly categorized into neural solvers and symbolic solvers. Neural solvers represent algorithms with neural networks, which are trained to produce high-quality solutions via supervised learning or reinforcement learning (Cappart et al., 2023; Bengio et al., 2020). In contrast, symbolic solvers frame combinatorial optimization as a code generation task, leveraging large language models (LLMs) to synthesize algorithms in formal languages (OpenAI, 2024; DeepSeek-AI, 2025). These approaches exploit the instruction-following and reasoning capabilities of LLMs and incorporate feedback from test cases to iteratively refine solutions (Romera-Paredes et al., 2023; Liu et al., 2024; Ye et al., 2024).

Despite recent advances in both neural and symbolic approaches, a fundamental question remains:

To what extent can ML-based solvers match or surpass state-of-the-art (SOTA) human-designed algorithms for real-world combinatorial optimization problems?

Existing benchmarks fall short in addressing this question due to several limitations: (i) **Scale:** Current evaluations primarily focus on small-scale problem instances, such as graphs with fewer than one thousand nodes. While this simplifies benchmarking, it fails to reflect the complexity and scale of real-world CO tasks that can scale up to millions nodes (Leskovec & Krevl, 2014). On existing benchmarks, we observed that SOTA human-designed problem-specific solvers can find near-optimal solutions within seconds, making the evaluation provide limited insight into the efficiency of different methods. (ii) **Realism:** Most previous evaluation datasets are synthetic—often randomly generated and i.i.d. with respect to the training distribution—failing to capture solver performance on real-world data. For instance, prior TSP evaluations (Kool et al., 2019; Sun & Yang, 2023; Luo et al., 2023) are typically conducted on graphs with the same size and structure as those used during training, whereas real-world problems exhibit greater diversity and irregularity. (iii) **Data Availability:** Many existing CO benchmarks lack sufficient training data, making them un-

[*]Equal contribution [1]School of Computer Science, Carnegie Mellon University. Correspondence to: Shengyu Feng <shengyuf@cs.cmu.edu>.

The second AI for MATH Workshop at the 42^{nd} International Conference on Machine Learning, Vancouver, Canada.

suitable for assessing data-driven approaches (Fan et al., 2024; Tang et al., 2025; Sun et al., 2025).

To enable meaningful and comprehensive evaluation of contemporary ML-based CO solvers, we introduce **FrontierCO**, a curated collection of challenging instances spanning eight classical CO problems across five diverse domains. Unlike benchmarks solely composed of synthetic data, FRONTIERCO draws from established repositories (e.g., TSPLib (Reinelt, 1991), CVRPLib (CVR, 2014), SteinLib (Koch et al., 2000)), major competitions (e.g., the DIMACS Challenge (Johnson & McGeoch, 1993) and PACE Challenge (PAC, 2025)), and evaluation sets used by recent SOTA solvers (Naderi & Roshanaei, 2021; Gnägi & Baumann, 2021), ensuring both practical relevance and scientific rigor. We intentionally select instances that either cannot be efficiently solved by the best existing human-designed algorithms or lack known optimal solutions, thereby enabling performance assessment at the cutting edge of combinatorial optimization. For the problems without training data, we manually create script for generating diverse synthetic data to support training.

For each task in FRONTIERCO, we construct two test sets of varying difficulty: *easy set*, which contains instances historically considered challenging but now solvable by SOTA methods; and *hard set*, which focused on open, hard instances that remain unsolved or computationally intensive. The first set is mainly used to validate the effectiveness of ML-based solvers, while the second set, which is free from any possible human heuristic hacking, serves as the main evaluation set in testing whether the ML-based solvers truly advance the human intelligence. This design enables a comprehensive evaluation of different application scenarios of CO solvers.

Using FRONTIERCO, we conduct a systematic evaluation of a broad spectrum of ML-based CO solvers, including: end-to-end neural solvers, neural-enhanced traditional algorithms, and LLM-powered agentic approaches. Our empirical study covers **13 recently neural solvers** and **3 SOTA LLM-based agents**, yielding the following key insights:

1. **Performance gap between human-designed SOTA solvers and ML-based solvers:** Despite recent progress in machine learning for combinatorial optimization, our evaluation reveals a substantial and persistent performance gap between ML-based solvers and human-designed SOTA solvers across all problem types and difficulty levels. This gap is notably larger than those reported in prior studies, primarily due to the inclusion of harder instances and more challenging training/testing conditions in our benchmark.

2. **Neural solvers exhibit significant performance degradation on hard problems:** This decline is primarily attributed to challenges in scaling when parameter spaces become substantially large, when problem structures grow more complex, or when the test-set instance sizes differ significantly from those in the training set. These limitations hinder the generalization capabilities of neural models. Consequently, addressing training-testing discrepancies and integrating neural solvers with powerful heuristic methods emerge as promising yet underexplored directions, offering viable alternatives to purely end-to-end neural approaches in combinatorial optimization.

3. **LLM-based agentic solvers exhibit significant performance variability:** Unlike traditional neural solvers, LLM-based agents possess the capability to invoke existing algorithms, including SOTA algorithms, to address combinatorial optimization problems. In principle, if an LLM agent can accurately identify and apply the appropriate SOTA solver for a given instance, it could achieve a primal gap of zero. However, our experiments reveal that LLM-based solvers often underperform compared to SOTA solvers and exhibit considerable variance in performance across different problem types and instances. This variability is primarily due to challenges in accurately matching problem instances to suitable algorithms and effectively integrating them. Enhancing the ability of LLM-based agents to recognize and leverage appropriate algorithms remains a promising yet underexplored research direction in combinatorial optimization.

Our evaluation also reveals promising directions for future development. LLM-based solvers outperform SOTA human-designed algorithms on the Maximum Independent Set and Capacitated Vehicle Routing problems, highlighting their ability to discover novel strategies. Neural solvers perform well on problems like Flexible Job-shop Scheduling, where a global receptive field helps capture long-range structure. These findings suggest that combining neural models with LLM-discovered heuristics may offer a powerful path forward, uniting strong low-level representations with flexible high-level reasoning.

The contributions of our work are summarized as follows:

- **Introduction of FrontierCO:** We present FRONTIERCO, the most comprehensive and challenging benchmark to date for evaluating CO solvers. It encompasses a diverse array of real-world CO problems, provides curated datasets suitable for training ML models, and includes large-scale instances to facilitate scalability analysis.

- **Comprehensive Evaluation of ML-Based CO Methods:** We conduct a systematic evaluation of 16 contemporary ML-based CO methods, including end-to-end neural solvers, neural-augmented traditional algorithms, and LLM-based symbolic solvers, comparing them directly with SOTA human-designed solvers within a unified and rigorous experimental framework.

- **Insights and Future Directions:** Our findings offer critical insights into the comparative strengths and limitations of neural and LLM-based agentic solvers. They also highlight promising avenues for future research and establish clear metrics and evaluation protocols to guide ongoing efforts at the intersection of machine learning and combinatorial optimization.

2. FrontierCO: the Proposed Benchmark

2.1. Formal Objective and Evaluation Metrics

We follow (Papadimitriou & Steiglitz, 1982; Sun & Yang, 2023) in denoting a combinatorial optimization (CO) problem instance as s, a solution as $x \in \mathcal{X}_s$, and defining the cost function $c_s : \mathcal{X}_s \to \mathbb{R}_+$ as

$$c_s(x) = \text{cost}(x; s) + \text{valid}(x; s), \tag{1}$$

where $\text{cost}(x; s)$ is a problem-specific objective assumed to be non-negative in this work (e.g., the tour length in Traveling Salesman Problem or the subset size in Maximum Independent Set), and $\text{valid}(x; s)$ penalizes constraint violations—taking value ∞ if x is infeasible for instance s, and 0 otherwise.

To accommodate the varying scales of different problem instances, we normalize the cost via the following definition:

$$\text{pg}(x; s) = \begin{cases} 1, & \text{if } x \text{ is infeasible,} \\ \frac{|\text{cost}(x;s) - c^*|}{\max\{|\text{cost}(x;s)|, |c^*|\}}, & \text{otherwise,} \end{cases} \tag{2}$$

where c^* is the (precomputed) optimal or best-known cost for instance s, and $\text{pg}(x; s)$ denotes the *primal gap* (Berthold, 2006) of x with respect to c^*.

Let A denote a specific algorithm in the search space $\mathcal{A}$, and let D be a distribution over problem instances. The objective of algorithm search is then defined as:

$$\min_{A \in \mathcal{A}} \mathbb{E}_{s \sim D,\, x \sim A} [\, \text{pg}(x; s)\,]. \tag{3}$$

The search space $\mathcal{A}$ includes all possible parameterizations of neural solvers or all feasible token sequences generated by symbolic solvers, depending on the solver type.

For comparative evaluation, we report the test-set average primal gap achieved by each CO solver, along with the corresponding inference time (in seconds) required to obtain that result. Note that we do not compare training time, a standard practice in CO evaluations, since classical non-ML solvers (exact or heuristic) do not involve a training phase. These solvers are based instead on hand-crafted human knowledge and algorithmic heuristics. In contrast, ML-based solvers require data-driven training but do not depend on manually designed heuristics.

2.2. Domain Coverage

This study focuses on eight types of combinatorial optimization (CO) problems that have gained increasing attention in recent machine learning research due to their practical significance and theoretical importance in the CO domain. These problems are:

- **MIS (Maximum Independent Set)**: Find the largest subset of non-adjacent vertices in a graph.
- **MDS (Minimum Dominating Set)**: Find the smallest subset of vertices such that every vertex in the graph is either in the subset or adjacent to a vertex in the subset.
- **TSP (Traveling Salesman Problem)**: Find the shortest possible tour that visits each city exactly once and returns to the starting point. We focus on the 2D Euclidean space in this work.
- **CVRP (Capacitated Vehicle Routing Problem)**: Determine the optimal set of delivery routes for a fleet of vehicles with limited capacity to serve a set of customers.
- **CFLP (Capacitated Facility Location Problem)**: Choose facility locations and assign clients to them to minimize the total cost, subject to facility capacity constraints.
- **CPMP (Capacitated *p*-Median Problem)**: Select p facility locations and assign clients to them to minimize the total distance, while ensuring that no facility exceeds its capacity.
- **FJSP (Flexible Job-Shop Scheduling Problem)**: Schedule a set of jobs on machines where each operation can be processed by multiple machines, aiming to minimize the makespan while respecting job precedence and machine constraints.
- **STP (Steiner Tree Problem)**: Find a minimum-cost tree that spans a given subset of terminal nodes in a graph, possibly including additional intermediate nodes known as Steiner points (Maculan, 1987). Note that we consider the general *STP in graphs* rather than Euclidean STP (Beasley, 1992).

The dataset statistics are summarized in Table 1, with additional details provided in the Appendix A. Note that only test data are collected from the listed sources; training and validation data are regenerated by us to eliminate inconsistencies found in previous evaluations (see Section 2.4).

Graph-based problems (MIS and MDS) and routing problems (TSP and CVRP) have been widely used to evaluate end-to-end neural solvers (Qiu et al., 2022; Zhang et al., 2023; Sun & Yang, 2023; Sanokowski et al., 2025), as these tasks often admit relatively straightforward decoding strategies to transform probabilistic model output into feasible solutions. In contrast, facility location and scheduling problems (such as CFLP, CPMP, and JSSP) involve more complex and interdependent constraints, making them better suited to hybrid approaches that combine neural networks with traditional solvers (Gasse et al., 2019; Scavuzzo et al., 2022; Feng & Yang, 2025b). Tree-based problems have received comparatively less attention in neural combinatorial optimization, yet we include a representative case (e.g., STP) due to their fundamental importance in the broader CO landscape. All of the above problems can also be directly handled by symbolic solvers, enabling comprehensive and comparable evaluations across solver paradigms (Romera-Paredes et al., 2023; Liu et al., 2024; Ye et al., 2024).

2.3. Problem Instances

For each CO problem type, we collect a diverse pool of problem instances from problem-specific and comprehensive CO libraries (Reinelt, 1991; Koch et al., 2000; CVR, 2014; Beasley, 1990), major CO competitions (Johnson & McGeoch, 1993; PAC, 2025), and evaluation sets reported in recent research papers. These instances are available at `https://huggingface.co/datasets/CO-Bench/FrontierCO`.

Due to rapid progress in combinatorial optimization, many instances from earlier archives can now be effectively solved by state-of-the-art (SOTA) problem-specific solvers, often achieving an optimality gap below 1% within a 1-hour time budget. We select a representative subset of such instances as our *easy set*, which serves to validate the baseline effectiveness of ML-based solvers.

To evaluate performance under more realistic and demanding conditions, we also construct a *hard set* comprising open benchmark instances widely used to assess cutting-edge human-designed algorithms. Many of these instances lack known optimal solutions and remain beyond the reach of existing heuristics. As a result, they are less susceptible to *heuristic hacking*—a phenomenon where neural solvers or LLM-based agents rely on handcrafted decoding strategies or memorize prior solutions, rather than learning to solve the problem from first principles.

Importantly, our hard set is not defined merely by instance size, as is common in prior work. Instead, we emphasize structurally complex cases, such as hypercube graphs in STP (Rosseti et al., 2001) or SAT-induced MIS (Xu et al., 2007), which require models to understand and reason about intricate problem structures.

2.4. SOTA Solvers and Best Known Solutions (BKS)

We identify the state-of-the-art (SOTA) solver for each CO problem type based on published research papers and competition leaderboards. The selected solvers include: KaMIS (Lamm et al., 2017) for MIS, LKH-3 (Helsgaun, 2017) for TSP, HGS (Vidal et al., 2012) for CVRP, GB21-MH (Gnägi & Baumann, 2021)—a hybrid metaheuristic—for CPMP, and SCIP-Jack (Rehfeldt et al., 2021) for STP. For problems where no dominant problem-specific solver is available (e.g., MDS, CFLP, FJSP), we rely on general-purpose commercial solvers, such as Gurobi (Gurobi Optimization, LLC, 2024) for MDS and CFLP (Mixed Integer Programming), and CPLEX (Cplex, 2009) for FJSP (Constraint Programming).

All of these solvers fall into the category of manually designed heuristics or exact solvers. Notably, ML-based methods have not yet surpassed the best-performing human-designed solvers in the CO domain. Among them, Gurobi, CPLEX and SCIP-Jack are exact solvers; the rest are heuristic-based.

Prior evaluations of ML-based CO solvers often relied on self-generated synthetic test instances, leading to difficulties in fair comparison across papers. These instances are sensitive to implementation details such as random seeds and Python versions, introducing undesirable variability and inconsistency. To address this, we provide standardized BKS for all test-set instances in our benchmark. These BKS are collected from published literature and competition leaderboards, and are further validated using the corresponding SOTA solvers executed on our servers. For instances lacking known BKS, such as the MDS instances from the ongoing PACE Challenge 2025 (PAC, 2025), or for benchmarks with outdated references, such as those in the CFLP literature (Avella & Boccia, 2009; Avella et al., 2009), we run the designated SOTA solver for up to two hours to obtain high-quality reference solutions.

2.5. Standardized Training/Validation Data

Similar to BKS, inconsistencies in self-generated training and validation data can also contribute to difficulties in cross-paper comparisons. To address this, FrontierCO provides standardized training sets for neural solvers and development sets for LLM agents, generated using a variety of problem-specific instance generators (see Appendix A).

We also release a complete toolkit that includes a data loader, an evaluation function, and an abstract solving template tailored for LLM-based agents. The data loader and evaluation function are hidden from the agents to prevent data leakage. The solving template provides a natural language problem description along with Python starter code specifying the expected input and output formats. An example prompt is provided in Appendix C.

Table 1. Summary of Collected Problem Instances

Problem	Test Set Sources	Attributes	Easy Set	Hard set
MIS	2nd DIMACS Challenge (Johnson & Trick, 1996) BHOSLIB (Xu et al., 2007)	Instances Nodes	36 1,404–7,995,464	16 1,150–4,000
MDS	PACE Challenge 2025 (PAC, 2025)	Instances Nodes	20 2,671–675,952	20 1,053,686–4,298,062
TSP	TSPLib (Reinelt, 1991) 8th DIMACS Challenge (Johnson & McGeoch, 2007)	Instances Cities	29 1,002–18,512	19 10,000–10,000,000
CVRP	Golden et al. (Golden et al., 1998) Arnold et al. (Arnold et al., 2019) CVRPLib (CVR, 2014) 12th DIMACS Challenge (DIM, 2021-2022)	Instances Cities Min. Vehicles	20 200–483 5–38	10 3000–30000 46–512
CFLP	Avella and Boccia (Avella & Boccia, 2009) Avella et al. (Avella et al., 2009)	Instances Facilities Customers	20 1,000 1,000	30 2,000 2,000
CPMP	Lorena and Senne (Lorena & Senne, 2000; 2004) Stefanello et al. (Stefanello et al., 2015) Gnägi and Baumann (Gnägi & Baumann, 2021)	Instances Facilities Medians	31 100–4,461 10–1,000	12 10,510–498,378 100–2,000
FJSP	Behnke and Geiger (Behnke & Geiger, 2012) Naderi and Roshanaei (Naderi & Roshanaei, 2021)	Instances Jobs Machines	60 10–100 10–20	20 10–100 20–60
STP	Vienna-GEO (Leitner et al., 2014) PUC (Rosseti et al., 2001) SteinLib (Koch et al., 2000) 11th DIMACS Challenge (DIM, 2013-2014)	Instances Nodes Terminals	23 7,565–71,184 86–6,107	50 64–4,096 8–2,048

3. Evaluation Design

3.1. Implementation Settings

To comprehensively evaluate solver performance, we allow a maximum solving time of one hour per problem instance. While some heuristic solvers may terminate earlier upon finding a suboptimal solution, the time budget ensures comparability across solver types.

For fair comparison, each solver is executed on a single CPU core of a dual AMD EPYC 7313 16-Core processor, and neural solvers are run on a single NVIDIA RTX A6000 GPU. *We use only the primal gap defined in Equation 2 as the evaluation metric.*

Solving time is reported for reference only, as it is influenced by factors such as compute hardware (CPU vs. GPU), solver type (exact vs. heuristic), and implementation language (C++ vs. Python). For any infeasible solution, we assign a primal gap of 1 and a solving time of 3600 seconds (i.e., the full time budget).

Note that the primal gap is computed relative to the best known solution (BKS), so its absolute value does not directly reflect the inherent difficulty of the instance—especially in cases where no known optimum exists. The arithmetic mean of the primal gaps and geometric mean of solving time are reported across our experiments.

3.2. Representative Neural Solvers

In addition to the SOTA human-designed solvers described in Section 2.4, we include a curated set of machine learning-based CO solvers from recent literature. Our focus is primarily on neural solvers evaluated on a subset of problems, complemented by a group of general-purpose LLM-based agentic solvers evaluated across all eight CO problem types.

The neural solvers are tailored to specific problems:

- **DiffUCO** (Sanokowski et al., 2024): An unsupervised diffusion-based neural solver for MIS and MDS that learns from the Lagrangian relaxation objective.
- **SDDS** (Sanokowski et al., 2025): A more scalable version of DiffUCO for MIS and MDS, with efficient training process.
- **RLNN** (Feng & Yang, 2025a): A Regularized Langevin Dynamics framework that enhances exploration in CO by enforcing expected distances between sampled and current solutions, leading to improved performance in both simulated annealing and neural network-based solvers.
- **LEHD** (Luo et al., 2023): A hybrid encoder-decoder model for TSP and CVRP, with strong generalization to real-world instances.
- **DIFUSCO** (Sun & Yang, 2023): A diffusion-based ap-

proach for TSP that achieves strong scalability, solving instances with up to 10,000 cities.

- **DeepACO** (Ye et al., 2023): A neural solver that adapts Ant Colony Optimization (ACO) principles to learn metaheuristic strategies for solving TSP and CVRP.
- **tMDP** (Scavuzzo et al., 2022): A reinforcement learning framework that models the branching process in Mixed Integer Program (MIP) solvers as a tree-structured Markov Decision Process.
- **SORREL** (Feng & Yang, 2025b): A reinforcement learning method that leverages suboptimal demonstrations and self-imitation learning to train branching policies in MIP solvers.
- **GCNN** (Gasse et al., 2019): A graph convolutional network (GNN)-guided solver for CFLP and CPMP, which learns to guide branching decisions within a branch-and-bound framework.
- **IL-LNS** (Sonnerat et al., 2021): A neural large neighborhood search method for Integer Linear Programs (ILPs) that incorporates learned policies into local search operators and uses SCIP (Achterberg, 2009) as the underlying MIP solver.
- **CL-LNS** (Huang et al., 2023): A contrastive learning-based large neighborhood search approach for ILPs that learns effective destroy heuristics by contrasting positive and negative solution samples, achieving state-of-the-art performance on several benchmarks.
- **MPGN** (Lei et al., 2022): A reinforcement learning-based approach for FJSP that employs multi-pointer graph networks to capture complex dependencies and generate efficient schedules.
- **L-RHO** (Li et al., 2025): A learning-guided rolling horizon optimization method that integrates machine learning predictions into the rolling horizon framework to improve decision-making for FJSP.

Since STP is not well studied by existing ML methods, we consider vanilla reinforcement learning (**RL**) based on REINFORCE and supervised learning (**SL**) baselines, predicting the Steiner points. The Takahashi–Matsuyama algorithm (Takahashi & Matsuyama, 1980) is then applied to decode the prediction into a valid Steiner tree.

3.3. Representative LLM-based Agentic Solvers

Our LLM-based solvers are selected based on the CO-Bench evaluation protocol (Sun et al., 2025), including both general-purpose prompting approaches and CO-specific iterative strategies:

- **FunSearch** (Romera-Paredes et al., 2023): An evolutionary search framework that iteratively explores the solution space and refines candidates through backtracking and pruning.
- **Self-Refine** (Madaan et al., 2023; Shinn et al., 2023): A feedback-driven refinement method in which the LLM improves its own output via iterative self-refinement.
- **ReEvo** (Shinn et al., 2023): A self-evolving agent that leverages past trajectories to refine its future decisions through reflective reasoning.

Unlike the neural solvers, which are applied to problem-specific subsets, all LLM-based solvers are evaluated across the full set of eight CO problem types in our benchmark.

4. Results

We summarize the comparative results in Figures 1. We also report the run-time statistics for the SOTA solvers in Figure 2. Detailed results are available in Appendix B.

We draw several key observations from our results. **First, there is a substantial performance gap on average between human-designed state-of-the-art (SOTA) solvers and ML-based solvers** across all types of problems and difficulty levels. Notably, this gap is more pronounced in our benchmark compared to previously published results. For example, LEHD exhibits a reported 0.72% gap on a standard TSP benchmark (Kool et al., 2019), whereas on our new benchmark, the gap expands to 10% in easy TSP instances and a striking 77% in hard instances. One major contributing factor to this discrepancy is the difference in the training and evaluation settings. In prior studies, neural solvers were typically trained on synthetic graphs of a fixed size (e.g. 10,000 nodes) and evaluated on test instances of the same size, resulting in well-aligned training and testing conditions. In contrast, our datasets introduce substantial variability in graph size and structure in both training and test sets. This setup better reflects real-world deployment scenarios, but also introduces significant challenges due to distribution shifts between training and test data. Consequently, LEHD and many other ML-based methods exhibit severe performance degradation in FrontierCO.

Second, neural solvers suffer significantly from scalability issues on hard problem instances. In particular, prior state-of-the-art neural solvers frequently encounter out-of-memory or timeout failures when applied to large-scale problems, resulting in unsuccessful runs and 100% primal gaps, for example, on MDS and CPMP. This underscores a critical practical limitation that has been largely overlooked in existing research on machine learning for combinatorial optimization, primarily due to the absence of realistic large-scale benchmarks.

Third, LLM-based agents outperform neural solvers but exhibit significant performance variability. LLM-based

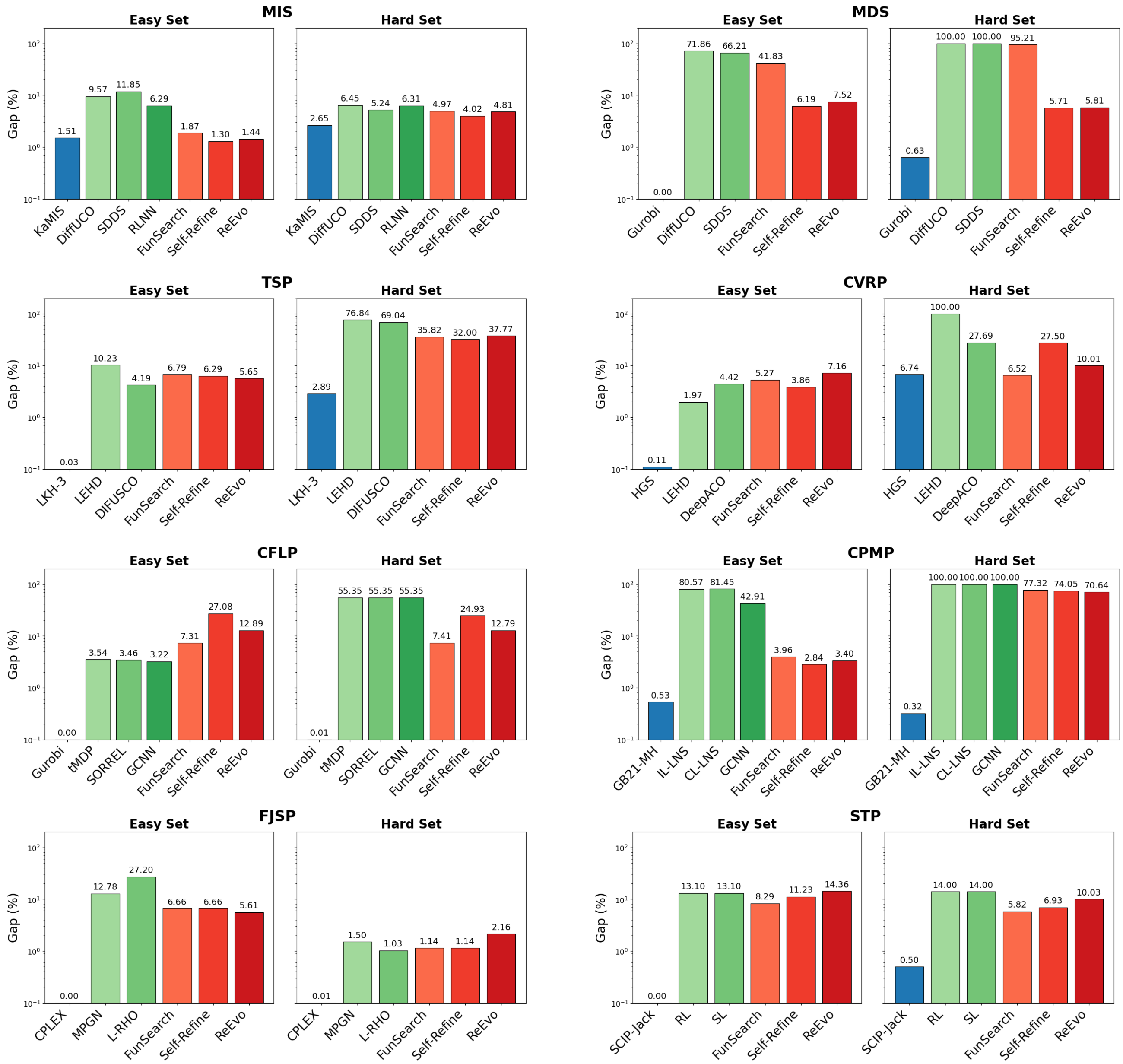

Figure 1. Gap (%) comparison of classical and ML-based solvers across eight CO problems with easy and hard test instances (lower is better). The classical solvers are in deep blue, neural solvers are colored in green, and LLM agentic solvers are represented in red.

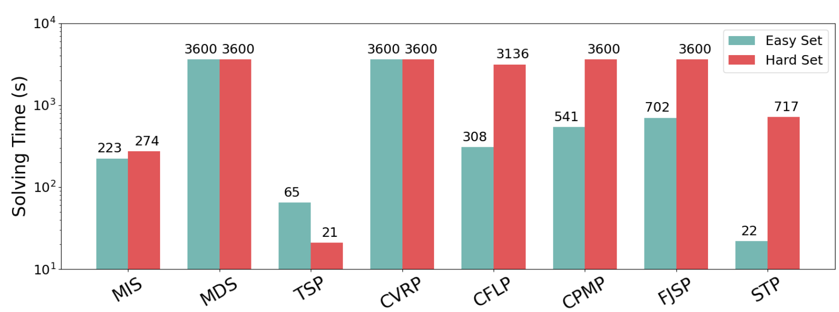

Figure 2. Solving time (in seconds) for the SOTA classical solvers on eight CO problems with easy and hard test sets, respectively.

Table 2. Neural Solvers with or without the Neural Module

TSP-Easy			CVRP-Easy			CFLP-Easy		
Method	Gap ↓	Time ↓	Method	Gap ↓	Time ↓	Method	Gap ↓	Time ↓
LKH-3	0.03%	65s	HGS	0.11%	3600s	Gurobi	0.00%	308s
2-OPT	20.09%	3600s	ACO	**4.22%**	54s	SCIP	6.50%	3600s
DIFUSCO	**4.19%**	555s	DeepACO	4.42%	50s	GCNN	**3.22%**	3600s

approaches such as FunSearch, Self-Refine, and ReEvo generally achieve stronger results than neural solvers in our evaluation, partly because they are less prone to out-of-memory failures and more resilient to distribution shifts. However, their performance varies substantially across the eight problems. For instance, while these methods perform comparably to the state-of-the-art solver HGS on the CVRP hard set, they fall significantly short on TSP—even though both are routing problems. In principle, LLMs possess the flexibility to invoke any external package and implement any algorithm, making them theoretically capable of achieving SOTA performance by selecting and executing the optimal solver for a given CO task. In practice, however, current LLM-based agents lack the robustness needed to consistently identify appropriate solvers and integrate them effectively, as reflected in our results. Enhancing the ability of LLM agents to recognize and leverage suitable algorithms remains a promising yet underexplored direction for combinatorial optimization.

Fourth, some ML-based solvers outperform prior SOTA human-designed solvers on certain problem sets. For example, Self-Refine outperforms KaMIS on the easy set of MIS, and FunSearch outperforms HGS on the hard set of CVRP. Upon examining the underlying algorithms, we find that Self-Refine employs a kernelization technique to simplify MIS instances and solves small kernels exactly using a Tomita-style max-clique algorithm. For larger instances, it applies an ARW-style heuristic with solution pools, crossover, and path-relinking. Similarly, on CVRP, FunSearch designs an Iterated Local Search framework enhanced with regret insertion and Variable Neighborhood Descent. These results demonstrate the strong potential of ML-based methods to automatically develop competitive or even superior solvers for frontier combinatorial optimization problems. We believe that it is a promising direction to extend the capabilities of LLMs to a broader range of problems and to further explore hybrid approaches that combine the strengths of neural solvers and LLM-based methods.

5. Discussions

5.1. Does the Neural Module Help?

Considering the performance gap between neural solvers and state-of-the-art (SOTA) solvers, a natural question arises: does the neural module actually contribute to improved performance? To explore this, we conduct an ablation study by removing the neural component (or replacing it with a human-designed heuristic) from the underlying algorithm of each neural solver. We evaluate three representative pairs: DIFUSCO (Sun & Yang, 2023) vs. (greedy) 2-OPT, DeepACO (Ye et al., 2023) vs. ant colony optimization (ACO), and GCNN (Gasse et al., 2019) vs. SCIP (Achterberg, 2009). The results are summarized in Table 2.

The results show that both DIFUSCO and GCNN significantly improve upon their respective heuristic baselines, indicating a meaningful contribution from the neural module. In contrast, DeepACO fails to outperform vanilla ACO, possibly due to the influence of the HGS (Vidal et al., 2012) post-processing used in its implementation. Overall, our findings suggest that neural components can enhance human-designed heuristics, but such improvements are typically realized when built on relatively weak algorithms. Whether similar gains can be achieved when enhancing strong heuristics remains unclear, especially for challenging problem instances requiring prolonged search processes.

5.2. Do Neural Solvers Capture Global Structure?

Most neural solvers are based on graph neural networks (GNNs), which rely on local message passing. While they have demonstrated strong performance on routing problems such as TSP and CVRP—which involve complex global constraints—the majority of existing evaluations are limited to 2D Euclidean instances. Compared to general graph problems, Euclidean instances—such as those in metric TSP—often exhibit favorable local structures (e.g., triangle inequality), which can be explicitly exploited by certain algorithms to achieve improved performance (Karlin et al., 2021). In contrast, graph problems such as MIS lack such spatial regularities, and neural solvers often perform poorly (Angelini & Ricci-Tersenghi, 2022; Böther et al., 2022).

To explicitly evaluate the ability of neural solvers in capturing global structure, we leverage the rich source of STP instances, which includes both Euclidean and non-Euclidean graphs (see Appendix A.8 for details). We train two separate GNNs to predict Steiner nodes, using ground truth labels generated by SCIP-Jack (Rehfeldt et al., 2021). One model is trained on Euclidean instances, and the other on non-Euclidean instances. The results are shown in Figure 3.

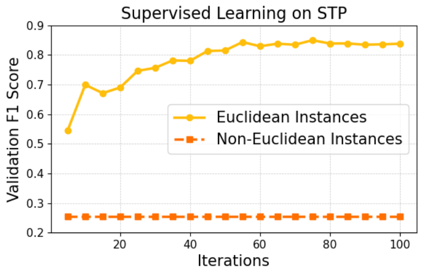

Figure 3. Training dynamics on Euclidean and non-Euclidean STP.

The results reveal a clear contrast: while the GNN quickly learns to predict Steiner points on Euclidean graphs, it fails to make any progress on non-Euclidean ones. This suggests that existing GNNs implicitly rely on locality and cannot capture the global structure required by many CO problems. These findings underscore a fundamental limitation in the expressive power of current neural solvers.

5.3. What Kinds of Algorithms Do LLMs Discover?

To better understand the algorithmic strategies developed by LLM-based solvers, we visualize the key words corresponding to their generated algorithms using the word cloud in Figure 4, where the size of each word reflects its frequency of appearance across algorithms.

Figure 4. Word cloud of the algorithms generated by LLMs.

A clear pattern emerges: classical metaheuristics, particularly simulated annealing (SA) and large neighborhood search (LNS), consistently appear across a diverse set of problems and often form the foundation of LLM-generated algorithms. This highlights a shared reliance on well-established CO algorithms that effectively balance exploration and exploitation. While current LLMs still fall short of demonstrating novel algorithmic reasoning in combinatorial optimization, their strategies tend to replicate known metaheuristics and problem-specific techniques from the literature. Interestingly, we observe that their performance does not critically depend on integrating existing solvers, suggesting that LLMs can autonomously construct plausible and often effective algorithms. This adaptability is particularly promising for rapidly tackling new problem variants or classical problems with additional constraints, indicating strong potential for LLMs in few-shot algorithm design.

6. Related Work

Current machine-learning approaches to combinatorial optimization fall into two broad categories: neural and symbolic solvers. Neural solvers primarily train a graph neural network (GNN) model with standard machine learning objectives (Bengio et al., 2020; Cappart et al., 2023). The trained GNN is then used either to predict complete solutions directly (Luo et al., 2023; Sun & Yang, 2023; Sanokowski et al., 2024; 2025) or to guide classical heuristics such as branch-and-bound, ant-colony optimization, or Langevin dynamics (Gasse et al., 2019; Sonnerat et al., 2021; Ye et al., 2023; Li et al., 2025; Feng & Yang, 2025a). Symbolic solvers instead attempt to generate executable programs that solve the problem, exploring the space of algorithmic primitives with reinforcement learning (Kuang et al., 2024a;b) or leveraging LLM agents for code generation (Romera-Paredes et al., 2023; Ye et al., 2024; Liu et al., 2024).

Despite these advances, empirical studies have mostly focused on small synthetic benchmarks (Kool et al., 2019; Zhang et al., 2023; Li et al., 2025), overlooking the scalability and generalization issues in real-world problems. Besides, the lack of training instances in existing LLM agentic benchmarks (Fan et al., 2024; Tang et al., 2025; Sun et al., 2025) also hinders the development. To bridge these gaps, we introduce a comprehensive benchmark with both realistic evaluation instances and diverse training data sources.

7. Conclusion

We present FRONTIERCO, a new benchmark designed to rigorously evaluate ML-based CO solvers under realistic, large-scale, and diverse problem settings. Through a unified empirical study, we reveal that while current ML methods—including both neural and LLM-based solvers—show potential, they still lag behind state-of-the-art human-designed algorithms in terms of efficiency, generalization, and scalability. However, our findings also uncover promising avenues: neural solvers excel on structured problems, and LLM agents demonstrate novel strategy discovery on hard instances. We hope FRONTIERCO will serve as a foundation for advancing the design and evaluation of next-generation ML-based CO solvers.

Limitations and Future Work FRONTIERCO relies on human-designed SOTA solvers to produce (near) optimal solutions for test set instances, which can be computationally expensive for large CO problems—particularly in academic settings with limited computational resources. However, the methodology presented in this work serves as a valuable blueprint for the broader community, especially those with industrial-scale resources, to follow in developing more challenging and realistic CO benchmarks in the future.

Impact Statement

FRONTIERCO offers a standardized, challenging benchmark to advance ML for combinatorial optimization. It enables rigorous, reproducible evaluation, encourages scalable and generalizable solver development, and highlights current limitations to guide more robust and impactful AI solutions in real-world decision-making.

References

11th DIMACS Implementation Challenge: Steiner Tree Problems. Website, 2013-2014. URL https://dimacs11.zib.de/. Co-organized by DIMACS and ICERM. Available at https://dimacs11.zib.de/.

Cvrplib: Capacitated vehicle routing problem library. Website, 2014. URL http://vrp.galgos.inf.puc-rio.br/index.php/en/. Available at http://vrp.galgos.inf.puc-rio.br/index.php/en/.

12th DIMACS Implementation Challenge: Vehicle Routing Problems. Website, 2021-2022. URL http://dimacs.rutgers.edu/programs/challenge/vrp/. Organized by Mauricio Resende, Eduardo Uchoa, et al. Available at http://dimacs.rutgers.edu/programs/challenge/vrp/.

Pace 2025 Challenge: Dominating Set. Website, 2025. URL https://pacechallenge.org/2025/ds/. Parameterized Algorithms and Computational Experiments Challenge. Available at https://pacechallenge.org/2025/ds/.

Achterberg, T. Scip: solving constraint integer programs. *Mathematical Programming Computation*, 1:1–41, 2009.

Almeida, G., Martins de Sá, E., Souza, S., and Souza, M. A hybrid iterated local search matheuristic for large-scale single source capacitated facility location problems. *Journal of Heuristics*, 30:1–28, 12 2023. doi: 10.1007/s10732-023-09524-9.

Angelini, M. C. and Ricci-Tersenghi, F. Modern graph neural networks do worse than classical greedy algorithms in solving combinatorial optimization problems like maximum independent set. *Nature Machine Intelligence*, 5(1):29–31, December 2022. ISSN 2522-5839. doi: 10.1038/s42256-022-00589-y. URL http://dx.doi.org/10.1038/s42256-022-00589-y.

Arnold, F., Gendreau, M., and Sörensen, K. Efficiently solving very large-scale routing problems. *Comput. Oper. Res.*, 107(C):32–42, July 2019. ISSN 0305-0548. doi: 10.1016/j.cor.2019.03.006. URL https://doi.org/10.1016/j.cor.2019.03.006.

Arora, S. and Barak, B. *Computational Complexity: A Modern Approach*. Cambridge University Press, USA, 1st edition, 2009. ISBN 0521424267.

Avella, P. and Boccia, M. A cutting plane algorithm for the capacitated facility location problem. *Computational Optimization and Applications*, 43(1):39–65, May 2009. doi: 10.1007/s10589-007-9125-x. URL https://ideas.repec.org/a/spr/coopap/v43y2009i1p39-65.html.

Avella, P., Boccia, M., Sforza, A., and Vasilyev, I. An effective heuristic for large-scale capacitated facility location problems. *Journal of Heuristics*, 15:597–615, 12 2009. doi: 10.1007/s10732-008-9078-y.

Bahman Naderi, Rubén Ruiz, V. R. Repository for mixed-integer programming versus constraint programming for shop scheduling problems: New results and outlook. 2023. doi: 10.5281/zenodo.7541223. URL https://github.com/INFORMSJoC/2021.0326.

Barabási, A.-L. and Albert, R. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999. doi: 10.1126/science.286.5439.509. URL https://www.science.org/doi/abs/10.1126/science.286.5439.509.

Beasley, J. A heuristic for euclidean and rectilinear steiner problems. *European Journal of Operational Research*, 58(2):284–292, 1992. ISSN 0377-2217. doi: https://doi.org/10.1016/0377-2217(92)90214-T. URL https://www.sciencedirect.com/science/article/pii/037722179290214T. Practical Combinatorial Optimization.

Beasley, J. E. Or-library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41:1069–1072, 1990.

Behnke, D. and Geiger, M. J. Test instances for the flexible job shop scheduling problem with work centers. 2012. URL https://api.semanticscholar.org/CorpusID:54531116.

Bengio, Y., Lodi, A., and Prouvost, A. Machine learning for combinatorial optimization: a methodological tour d'horizon, 2020.

Berthold, T. *Primal heuristics for mixed integer programs*. PhD thesis, Zuse Institute Berlin (ZIB), 2006.

Böther, M., Kißig, O., Taraz, M., Cohen, S., Seidel, K., and Friedrich, T. What's wrong with deep learning in tree search for combinatorial optimization. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=mk0HzdqY7i1.

Cappart, Q., Chételat, D., Khalil, E. B., Lodi, A., Morris, C., and Veličković, P. Combinatorial optimization and reasoning with graph neural networks. *Journal of Machine Learning Research*, 24(130):1–61, 2023.

Caserta, M. and and, S. V. A general corridor method-based approach for capacitated facility location. *International Journal of Production Research*, 58(13): 3855–3880, 2020. doi: 10.1080/00207543.2019.1636320. URL https://doi.org/10.1080/00207543.2019.1636320.

Cornuejols, G., Sridharan, R., and Thizy, J. A comparison of heuristics and relaxations for the capacitated plant location problem. *European Journal of Operational Research*, 50(3):280–297, 1991. ISSN 0377-2217. doi: https://doi.org/10.1016/0377-2217(91)90261-S. URL https://www.sciencedirect.com/science/article/pii/037722179190261S.

Cplex, I. I. V12. 1: User's manual for cplex. *International Business Machines Corporation*, 46(53):157, 2009.

Dauzère-Pérès, S., Ding, J., Shen, L., and Tamssaouet, K. The flexible job shop scheduling problem: A review. *European Journal of Operational Research*, 314(2):409–432, 2024. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2023.05.017. URL https://www.sciencedirect.com/science/article/pii/S037722172300382X.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *ArXiv*, abs/2501.12948, 2025.

Diaz, J. and Fernandez, E. Hybrid scatter search and path relinking for the capacitated p-median problem. *European Journal of Operational Research*, 169:570–585, 02 2006. doi: 10.1016/j.ejor.2004.08.016.

Fan, L., Hua, W., Li, L., Ling, H., and Zhang, Y. NPHardEval: Dynamic benchmark on reasoning ability of large language models via complexity classes. In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4092–4114, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.225.

Feng, S. and Yang, Y. Regularized langevin dynamics for combinatorial optimization. In *International conference on machine learning*. PMLR, 2025a.

Feng, S. and Yang, Y. Sorrel: Suboptimal-demonstration-guided reinforcement learning for learning to branch. In *The 39th Annual AAAI Conference on Artificial Intelligence*, 2025b.

Fu, Z.-H., Sun, S., Ren, J., Yu, T., Zhang, H., Liu, Y., Huang, L., Yan, X., and Lu, P. A hierarchical destroy and repair approach for solving very large-scale travelling salesman problem, 2023. URL https://arxiv.org/abs/2308.04639.

Gadegaard, S., Klose, A., and Nielsen, L. An improved cut-and-solve algorithm for the single-source capacitated facility location problem. *EURO Journal on Computational Optimization*, 6, 04 2017. doi: 10.1007/s13675-017-0084-4.

Gasse, M., Chételat, D., Ferroni, N., Charlin, L., and Lodi, A. Exact combinatorial optimization with graph convolutional neural networks. In *Advances in Neural Information Processing Systems 32*, 2019.

Gnägi, M. and Baumann, P. A matheuristic for large-scale capacitated clustering. *Computers & Operations Research*, pp. 105304, 2021.

Golden, B. L., Wasil, E. A., Kelly, J. P., and Chao, I.-M. The impact of metaheuristics on solving the vehicle routing problem: Algorithms, problem sets, and computational results. 1998. URL https://api.semanticscholar.org/CorpusID:61757468.

Guastaroba, G. and Speranza, M. Kernel search for the capacitated facility location problem. *Journal of Heuristics*, 18:1–41, 12 2012. doi: 10.1007/s10732-012-9212-8.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024. URL https://www.gurobi.com.

Helsgaun, K. An extension of the lin-kernighan-helsgaun tsp solver for constrained traveling salesman and vehicle routing problems, 12 2017.

Huang, T., Ferber, A., Tian, Y., Dilkina, B., and Steiner, B. Searching large neighborhoods for integer linear programs with contrastive learning. In *International conference on machine learning*. PMLR, 2023.

Johnson, D. J. and Trick, M. A. *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, Workshop, October 11-13, 1993*. American Mathematical Society, USA, 1996. ISBN 0821866095.

Johnson, D. S. and McGeoch, C. C. *Network Flows and Matching: First DIMACS Implementation Challenge*. American Mathematical Society, USA, 1993. ISBN 0821865986.

Johnson, D. S. and McGeoch, L. A. *Experimental Analysis of Heuristics for the STSP*, pp. 369–443. Springer US, Boston, MA, 2007. ISBN 978-0-306-48213-7. doi: 10.1007/0-306-48213-4_9. URL https://doi.org/10.1007/0-306-48213-4_9.

Juhl, D., Warme, D., Winter, P., and Zachariasen, M. The geosteiner software package for computing steiner trees in the plane: an updated computational study. *Mathematical Programming Computation*, 10, 02 2018. doi: 10.1007/s12532-018-0135-8.

Karlin, A. R., Klein, N., and Gharan, S. O. A (slightly) improved approximation algorithm for metric tsp. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, pp. 32–45, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380539. doi: 10.1145/3406325.3451009. URL https://doi.org/10.1145/3406325.3451009.

Kawano, Y. A reduction from an lwe problem to maximum independent set problems. *Scientific Reports*, 13, 05 2023. doi: 10.1038/s41598-023-34366-7.

Koch, T., Martin, A., and Voß, S. SteinLib: An updated library on steiner tree problems in graphs. Technical Report ZIB-Report 00-37, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Takustr. 7, Berlin, 2000. URL http://elib.zib.de/steinlib.

Kool, W., van Hoof, H., and Welling, M. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ByxBFsRqYm.

Korte, B. and Vygen, J. *Combinatorial Optimization: Theory and Algorithms*. Springer Publishing Company, Incorporated, 5th edition, 2012. ISBN 3642244874.

Kuang, Y., Wang, J., Liu, H., Zhu, F., Li, X., Zeng, J., HAO, J., Li, B., and Wu, F. Rethinking branching on exact combinatorial optimization solver: The first deep symbolic discovery framework. In *The Twelfth International Conference on Learning Representations*, 2024a.

Kuang, Y., Wang, J., Zhou, Y., Li, X., Zhu, F., Hao, J., and Wu, F. Towards general algorithm discovery for combinatorial optimization: Learning symbolic branching policy from bipartite graph. In Salakhutdinov, R., Kolter, Z., Heller, K., Weller, A., Oliver, N., Scarlett, J., and Berkenkamp, F. (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 25623–25641. PMLR, 21–27 Jul 2024b.

Lamm, S., Sanders, P., Schulz, C., Strash, D., and Werneck, R. F. Finding near-optimal independent sets at scale. *J. Heuristics*, 23(4):207–229, 2017. doi: 10.1007/s10732-017-9337-x. URL https://doi.org/10.1007/s10732-017-9337-x.

Lei, K., Guo, P., Zhao, W., Wang, Y., Qian, L., Meng, X., and Tang, L. A multi-action deep reinforcement learning framework for flexible job-shop scheduling problem. *Expert Systems with Applications*, 205:117796, 2022. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2022.117796. URL https://www.sciencedirect.com/science/article/pii/S0957417422010624.

Leitner, M., Ljubic, I., Luipersbeck, M., Prossegger, M., and Resch, M. New real-world instances for the steiner tree problem in graphs, 01 2014.

Leskovec, J. and Krevl, A. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

Leskovec, J., Kleinberg, J., and Faloutsos, C. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, pp. 177–187, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 159593135X. doi: 10.1145/1081870.1081893. URL https://doi.org/10.1145/1081870.1081893.

Li, S., Ouyang, W., Ma, Y., and Wu, C. Learning-guided rolling horizon optimization for long-horizon flexible job-shop scheduling. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Aly68Y5Es0.

Liu, F., Tong, X., Yuan, M., Lin, X., Luo, F., Wang, Z., Lu, Z., and Zhang, Q. Evolution of heuristics: Towards efficient automatic algorithm design using large language model. In *ICML*, 2024.

Lorena, L. A. and Senne, E. L. Local search heuristics for capacitated p-median problems. 08 2000.

Lorena, L. A. and Senne, E. L. A column generation approach to capacitated p-median problems. *Computers & Operations Research*, 31(6):863–876, 2004. ISSN 0305-0548. doi: https://doi.org/10.1016/S0305-0548(03)00039-X. URL https://www.sciencedirect.com/science/article/pii/S030505480300039X.

Luo, F., Lin, X., Liu, F., Zhang, Q., and Wang, Z. Neural combinatorial optimization with heavy decoder: Toward large scale generalization. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=RBI4oAbdpm.

Maculan, N. The steiner problem in graphs**this work was partially supported by conselho nacional de desenvolvimento cientifico e tecnologico, cnpq 300195-83 and by finep. In Martello, S., Laporte, G., Minoux, M., and Ribeiro, C. (eds.), *Surveys in Combinatorial Optimization*, volume 132 of *North-Holland Mathematics Studies*, pp. 185–211. North-Holland, 1987. doi: https://doi.org/10.1016/S0304-0208(08)73236-5. URL https://www.sciencedirect.com/science/article/pii/S0304020808732365.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., Welleck, S., Majumder, B. P., Gupta, S., Yazdanbakhsh, A., and Clark, P. Self-refine: Iterative refinement with self-feedback. *ArXiv*, abs/2303.17651, 2023.

Naderi, B. and Roshanaei, V. Critical-path-search logic-based benders decomposition approaches for flexible job shop scheduling. *INFORMS Journal on Optimization*, 4, 08 2021. doi: 10.1287/ijoo.2021.0056.

OpenAI. Openai o1 system card, 2024.

Osman, I. Capacitated clustering problems by hybrid simulated annealing and tabu search, international transactions in operational research, 1, 317-336. *International Transactions in Operational Research*, 1:317–336, 07 1994. doi: 10.1016/0969-6016(94)90032-9.

Papadimitriou, C. and Steiglitz, K. *Combinatorial Optimization: Algorithms and Complexity*, volume 32. 01 1982. ISBN 0-13-152462-3. doi: 10.1109/TASSP.1984.1164450.

Qiu, R., Sun, Z., and Yang, Y. DIMES: A differentiable meta solver for combinatorial optimization problems. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022.

Rehfeldt, D., Shinano, Y., and Koch, T. Scip-jack: An exact high performance solver for steiner tree problems in graphs and related problems. In Bock, H. G., Jäger, W., Kostina, E., and Phu, H. X. (eds.), *Modeling, Simulation and Optimization of Complex Processes HPSC 2018*, pp. 201–223, Cham, 2021. Springer International Publishing. ISBN 978-3-030-55240-4.

Reinelt, G. Tsplib - a traveling salesman problem library. *INFORMS J. Comput.*, 3(4):376–384, 1991. URL http://dblp.uni-trier.de/db/journals/informs/informs3.html#Reinelt91.

Romera-Paredes, B., Barekatain, M., Novikov, A., Balog, M., Kumar, M. P., Dupont, E., Ruiz, F. J. R., Ellenberg, J. S., Wang, P., Fawzi, O., Kohli, P., Fawzi, A., Grochow, J., Lodi, A., Mouret, J.-B., Ringer, T., and Yu, T. Mathematical discoveries from program search with large language models. *Nature*, 625:468 – 475, 2023.

Rosseti, I., Poggi, M., Ribeiro, C., Uchoa, E., and Werneck, R. New benchmark instances for the steiner problem in graphs. 08 2001. doi: 10.1007/978-1-4757-4137-7_28.

Sanokowski, S., Hochreiter, S., and Lehner, S. A diffusion model framework for unsupervised neural combinatorial optimization. In *ICML*, 2024. URL https://openreview.net/forum?id=AFfXlKFHXJ.

Sanokowski, S., Berghammer, W. F., Wang, H. P., Ennemoser, M., Hochreiter, S., and Lehner, S. Scalable discrete diffusion samplers: Combinatorial optimization and statistical physics. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=peNgxpbdxB.

Scavuzzo, L., Chen, F. Y., Chételat, D., Gasse, M., Lodi, A., Yorke-Smith, N., and Aardal, K. Learning to branch with tree MDPs. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=M4OllVd70mJ.

Shinn, N., Cassano, F., Labash, B., Gopinath, A., Narasimhan, K., and Yao, S. Reflexion: language agents with verbal reinforcement learning. In *Neural Information Processing Systems*, 2023.

Sonnerat, N., Wang, P., Ktena, I., Bartunov, S., and Nair, V. Learning a large neighborhood search algorithm for mixed integer programs. *ArXiv*, abs/2107.10201, 2021. URL https://api.semanticscholar.org/CorpusID:236154746.

Statistisches Bundesamt. Gemeinden in deutschland nach fläche, bevölkerung und postleitzahl am 31.03.2017 (1. quartal), 2017. URL https://www.destatis.de/DE/ZahlenFakten/LaenderRegionen/Regionales/Gemeindeverzeichnis/Administrativ/Archiv/GVAuszugQ/AuszugGV1QAktuell.xlsx?blob=publicationFile. Accessed: 20 September 2017.

Stefanello, F., de Araújo, O. C. B., and Müller, F. M. Matheuristics for the capacitated p-median problem. *International Transactions in Operational Research*, 22 (1):149–167, 2015. doi: https://doi.org/10.1111/itor.12103. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/itor.12103.

Steglich, M. A hybrid heuristic based on self-organising maps and binary linear programming techniques for the capacitated p-median problem. 06 2019. doi: 10.7148/2019-0267.

Sun, W., Feng, S., Li, S., and Yang, Y. Co-bench: Benchmarking language model agents in algorithm search for combinatorial optimization, 2025. URL https://arxiv.org/abs/2504.04310.

Sun, Z. and Yang, Y. DIFUSCO: Graph-based diffusion solvers for combinatorial optimization. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=JV8Ff0lgVV.

Takahashi, H. and Matsuyama, A. An approximate solution for the steiner problem in graphs. *Mathematica Japonica*, 24(6):573–577, 1980.

Tang, J., Zhang, Q., Li, Y., Chen, N., and Li, J. Grapharena: Evaluating and improving large language models on graph computation. In *International Conference on Learning Representations*, 2025.

Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N., and Rei, W. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, 60(3):611–624, 2012. doi: 10.1287/opre.1120.1048. URL https://doi.org/10.1287/opre.1120.1048.

Xu, K. and Li, W. Exact phase transitions in random constraint satisfaction problems. *J. Artif. Int. Res.*, 12(1): 93–103, March 2000. ISSN 1076-9757.

Xu, K., Boussemart, F., Hemery, F., and Lecoutre, C. Random constraint satisfaction: Easy generation of hard (satisfiable) instances. *Artificial Intelligence*, 171(8):514–534, 2007. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2007.04.001. URL https://www.sciencedirect.com/science/article/pii/S0004370207000653.

Ye, H., Wang, J., Cao, Z., Liang, H., and Li, Y. Deepaco: Neural-enhanced ant systems for combinatorial optimization. In *Advances in Neural Information Processing Systems*, 2023.

Ye, H., Wang, J., Cao, Z., Berto, F., Hua, C., Kim, H., Park, J., and Song, G. Reevo: Large language models as hyperheuristics with reflective evolution. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Zhang, D., Dai, H., Malkin, N., Courville, A., Bengio, Y., and Pan, L. Let the flows tell: Solving graph combinatorial problems with GFlownets. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=sTjW3JHs2V.

A. Data Collection Details

This section outlines the data collection process for all problems, covering both test and training/validation instances. Since the training instance generation for neural solvers varies significantly across methods, we omit low-level details such as the number of instances and parameter settings. Instead, we focus on describing the generation of the validation set (test cases used to provide feedback for iterative agent refinement) used for LLM-based solvers.

A.1. Maximum Independent Set

To construct suitable test instances, we conduct a comprehensive re-evaluation of the datasets collected by Böther et al. (Böther et al., 2022). We find that some large real-world graphs (Leskovec & Krevl, 2014), such as ai-caida (Leskovec et al., 2005) with up to 26,475 nodes, are not particularly challenging for SOTA classical solvers like KaMIS (Lamm et al., 2017), which can solve them within seconds. Therefore, we select two moderately sized but more challenging datasets.

The easy test set comprises complementary graphs of the maximum clique instances from the 2nd DIMACS Challenge (Johnson & Trick, 1996), while the hard test set consists of the largest 16 instances (each with over 1,000 nodes) from the BHOSLIB benchmark (Xu et al., 2007), derived from SAT reductions. Since the original links have expired, we obtain these instances and their BKS from a curated mirror[1]. For those interested in additional sources of high-quality MIS instances, we also highlight vertex cover instances from the 2019 PACE Challenge[2], reductions from coding theory[3], and recent constructions derived from learning-with-errors (LWE) (Kawano, 2023), which provide a promising strategy for generating challenging MIS instances.

Training instances are generated using the RB model (Xu & Li, 2000), widely adopted in recent neural MIS solvers (Zhang et al., 2023; Sanokowski et al., 2024; 2025). We synthesize 20 instances with 800–1,200 nodes for our LLM validation set.

A.2. Minimum Dominating Set

Despite the popularity of MDS in evaluating neural solvers (Zhang et al., 2023; Sanokowski et al., 2024; 2025), we find a lack of high-quality publicly available benchmarks. We therefore rely on the PACE Challenge 2025[4], using the exact track instances as our easy set and the heuristic track instances as the hard set. From each, we selected the 20 instances with the highest primal-dual gaps after a one-hour run with Gurobi. Reference BKS are obtained by extending the solving time to two hours.

Training instances are Barabási–Albert graphs (Barabási & Albert, 1999) with 800–1,200 nodes, consistent with previous literature (Zhang et al., 2023; Sanokowski et al., 2024; 2025). We generate 20 such instances for the LLM validation set.

A.3. Traveling Salesman Problem

We source TSP instances from the 8th DIMACS Challenge[5] and TSPLib[6]. The easy test set includes symmetric 2D Euclidean TSP instances (distance type `EUC_2D`, rounding applied) from TSPLib with over 1,000 cities, all with known optimal solutions. This aligns with settings used in prior neural TSP solvers (Karlin et al., 2021).

The hard test set consists of synthetic instances from the DIMACS Challenge with at least 10,000 cities (Fu et al., 2023). We obtain BKS from the LKH website[7].

Training instances follow the standard practice of uniformly sampling points in a unit square (Kool et al., 2019). For simplicity, we reuse DIMACS instances with 1,000 nodes as our LLM training set, since they are drawn from the same distribution, except scaling the coordinates by a constant.

[1] https://iridia.ulb.ac.be/~fmascia/maximum_clique/
[2] https://pacechallenge.org/2019/
[3] https://oeis.org/A265032/a265032.html
[4] https://pacechallenge.org/2025/
[5] http://archive.dimacs.rutgers.edu/Challenges/TSP/
[6] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/
[7] http://webhotel4.ruc.dk/~keld/research/LKH/DIMACS_results.html

A.4. Capacitated Vehicle Routing Problem

We collect CVRP instances from the 12th DIMACS Challenge[8] and CVRPLib[9], which have significant overlap. From these, we select the Golden (collected by Arnold et al. (Golden et al., 1998)) and Belgium (collected by Arnold et al. (Arnold et al., 2019)) instances as our easy and hard sets, respectively. All BKS are retrieved from the CVRPLib website.

Training data generation follows the method used in DeepACO (Ye et al., 2023). Each instance includes up to 500 cities, with demands in [1, 9] and capacity fixed at 50. We generate 15 total validation instances for LLMs, with 5 each for 20, 100, and 500 cities.

A.5. Capacitated Facility Location Problem

Following the benchmark setup in previous works (Guastaroba & Speranza, 2012; Caserta & and, 2020), we select instances from Test Bed 1 (Avella & Boccia, 2009) and Test Bed B (Avella et al., 2009) as our easy and hard test sets, respectively. The easy set includes the 20 largest instances from Test Bed 1, each with 1,000 facilities and 1,000 customers. The hard set consists of the 30 largest instances from Test Bed B, each with 2,000 facilities and 2,000 customers. All instances are downloaded from the OR-Brescia website[10].

Notably, our easy instances are already significantly larger than the most challenging instances typically used in neural solver evaluations (Gasse et al., 2019; Scavuzzo et al., 2022; Feng & Yang, 2025b), which contain at most 100 facilities and 400 customers. All easy instances can be solved exactly by Gurobi. For the hard instances, as all available BKS identified in the literature (Caserta & and, 2020) are inferior to those obtained by Gurobi, we rerun Gurobi for two hours to obtain improved reference solutions.

Overall, we find that Gurobi already demonstrates strong performance on standard CFLP variants, in which each customer may be served by multiple facilities. Consequently, the single-source CFLP variant—where each customer must be assigned to exactly one facility—has become a more compelling and actively studied problem in recent CO literature (Gadegaard et al., 2017; Caserta & and, 2020; Almeida et al., 2023). Several corresponding benchmarks are also available on the OR-Brescia website.

For training data, we adopt the synthetic generation method from Cornuejols et al. (Cornuejols et al., 1991), producing 20 instances with 100 facilities and 100 customers for LLM validation. This generation method is widely used in existing neural branching works (Gasse et al., 2019; Scavuzzo et al., 2022; Feng & Yang, 2025b), and forms part of the construction for Test Bed 1 (Avella & Boccia, 2009).

A.6. Capacitated p-Median Problem

We follow the evaluation setup in recent works on CRMP (Stefanello et al., 2015; Gnägi & Baumann, 2021). Instances with fewer than 10,000 facilities are assigned to the easy set; larger ones go to the hard set. Easy instances include 6 real-world São José dos Campos instances (Lorena & Senne, 2004) and 25 adapted TSPLib instances (Lorena & Senne, 2000; Stefanello et al., 2015). These are sourced from INPE[11] and SomAla[12] websites. Hard instances are large-scale problems introduced by Gnägi and Baumann (Gnägi & Baumann, 2021), downloaded from their GitHub[13]. BKS are derived by combining the best GB21-MH results and values reported in (Stefanello et al., 2015; Steglich, 2019; Gnägi & Baumann, 2021).

In total, we collect 31 easy and 12 hard instances, all using Euclidean distances. Additional alternatives include spherical-distance instances (Diaz & Fernandez, 2006; Statistisches Bundesamt, 2017) and high-dimensional instances (Gnägi & Baumann, 2021).

We synthesize training data with Osman's method (Osman, 1994). The validation set for LLMs are generated by fixing the number of facilities at 500 and varying terminals in $\{5, 10, 20, 50\}$. Each setting includes 5 instances.

[8] http://dimacs.rutgers.edu/programs/challenge/vrp/cvrp/
[9] http://vrp.galgos.inf.puc-rio.br/index.php/en/
[10] https://or-brescia.unibs.it/home
[11] http://www.lac.inpe.br/~lorena/instancias.html
[12] http://stegger.net/somala/index.html
[13] https://github.com/phil85/GB21-MH

A.7. Flexible Job-Shop Scheduling Problem

We collect FJSP instances from two recent benchmark sets commonly used in the evaluation of classical FJSP solvers. The easy test set consists of instances introduced by Behnke and Geiger (Behnke & Geiger, 2012), available via a GitHub mirror[14]. The hard test set includes 24 of the largest instances (with 100 jobs) from a benchmark proposed by Naderi and Roshanaei (Naderi & Roshanaei, 2021), which we obtain from the official repository[15]. These two datasets are selected based on recent comparative studies in the literature (Bahman Naderi, 2023; Dauzère-Pérès et al., 2024).

Based on our literature review, the strongest results have been reported by the CP-based Benders decomposition method (Naderi & Roshanaei, 2021); however, the source code is not publicly available. As a result, we adopt a constraint programming approach using CPLEX, which has demonstrated consistently strong performance relative to other commercial solvers and heuristic methods (Bahman Naderi, 2023).

Training data is generated following the same protocol used in Li et al. (Li et al., 2025). Specifically, we synthesize 20 instances, each with 20 machines and 10 jobs, to form the LLM validation set.

A.8. Steiner Tree Problem

We collect STP instances from SteinLib[16] and the 11th DIMACS Challenge[17]. The easy set includes Vienna-GEO instances (Leitner et al., 2014), which—despite having tens of thousands of nodes—are solvable within minutes by SCIP-Jack. The hard set comprises PUC instances (Rosseti et al., 2001), most of which cannot be solved within one hour by SCIP-Jack and even lack known optima. BKS are determined by taking the best value between SCIP-Jack's one-hour primal bound and published solutions from SteinLib or Vienna-GEO (Leitner et al., 2014). We also highlight the 2018 PACE Challenge[18] as a useful benchmark with varied difficulty levels.

Training data includes two generation strategies. The first generator corresponds to the hardest instances in PUC (Rosseti et al., 2001), which constructs graphs from hypercubes with randomly sampled (perturbed) edge weights. We generate 100 training instances for neural solvers and 10 validation instances for LLMs across dimensions 6–10. The second, based on GeoSteiner (Juhl et al., 2018), samples 25,000-node graphs from a unit square. We include 15 such instances (10 for neural solvers, 5 for LLMs)[19], and add 45 adapted TSPLib instances (Juhl et al., 2018) to the neural training set. The LLM training set also serves as the validation set for neural solvers.

B. Detailed Results

Tables 3, 4, 5, and 6 present the detailed results for the evaluated methods in Section 4. A result is marked with * if the method suffers from the out-of-memory or timeout issue before obtaining a feasible solution on any instance in this benchmark.

Table 3. Comparative Results on MIS and MDS

MIS	Easy		Hard		**MDS**	Easy		Hard	
Method	Gap ↓	Time ↓	Gap ↓	Time ↓	Method	Gap ↓	Time	Gap ↓	Time ↓ ↓
KaMIS	1.51%	223s	**2.65%**	274s	Gurobi	**0.00%**	3600s	**0.63%**	3600s
DiffUCO	9.57%	154s	6.45%	19s	DiffUCO	71.86%	54s	100.00%*	3600s*
SDDS	11.85%	223s	5.24%	27s	SDDS	66.21%	54s	100.00%*	3600s*
RLNN	6.29%	532s	6.31%	1064s	RLNN	–	–	–	–
FunSearch	1.87%	3600s	4.97%	3600s	FunSearch	41.83%	3600s	95.21%	3600s
Self-Refine	**1.30%**	3600s	4.02%	3600s	Self-Refine	6.19%	3600s	5.71%	3600s
ReEvo	1.44%	3600s	4.81%	3600s	ReEvo	7.52%	3600s	5.81%	3600s

[14] https://github.com/Lei-Kun/FJSP-benchmarks
[15] https://github.com/INFORMSJoC/2021.0326
[16] https://steinlib.zib.de/steinlib.php
[17] https://dimacs11.zib.de/organization.html
[18] https://github.com/PACE-challenge/SteinerTree-PACE-2018-instances
[19] http://www.geosteiner.com/instances/

Table 4. Comparative Results on TSP and CVRP

TSP	Easy		Hard		**CVRP**	Easy		Hard	
Method	Gap ↓	Time ↓	Gap ↓	Time ↓	Method	Gap ↓	Time ↓	Gap ↓	Time ↓
LKH-3	**0.03%**	65s	2.89%	21s	HGS	**0.11%**	3600s	6.74%	3600s
LEHD	10.23%	487s	76.84%*	1347s*	LEHD	1.97%	893s	100.00%*	3600s*
DIFUSCO	4.19%	555s	69.04%*	2850s*	DeepACO	4.42%	50s	27.69%*	3333s*
FunSearch	6.79%	3600s	35.82%	3600s	FunSearch	5.27%	3600s	**6.52%**	3600s
Self-Refine	6.29%	3600s	32.00%	3600s	Self-Refine	3.86%	3600s	27.50%	3600s
ReEvo	5.65%	3600s	37.77%	3600s	ReEvo	7.16%	3600s	10.01%	3600s

Table 5. Comparative Results on CFLP and CPMP

CFLP	Easy		Hard		**CPMP**	Easy		Hard	
Method	Gap ↓	Time ↓	Gap ↓	Time ↓	Method	Gap ↓	Time ↓	Gap ↓	Time ↓
Gurobi	**0.00%**	308s	**0.01%**	3136s	GB21-MH	**0.53%**	541s	**0.32%**	3600s
tMDP	3.54%	3581s	55.35%	3600s	IL-LNS	80.57%*	3600s*	100.00%*	3600s*
SORREL	3.46%	3600s	55.35%	3600s	CL-LNS	81.45%*	3600s*	100.00%*	3600s*
GCNN	3.22%	3551s	55.35%	3600s	GCNN	42.91%*	2143s*	100.00%*	3600s*
FunSearch	7.31%	3600s	7.41%	3600s	FunSearch	3.96%	3600s	77.32%*	3600s*
Self-Refine	27.08%	3600s	24.93%	3600s	Self-Refine	2.84%	3600s	74.05%*	3600s*
ReEvo	12.89%	3600s	12.79%	3600s	ReEvo	3.40%	3600s	70.64%*	3600s*

Table 6. Comparative Results on FJSP and STP

FJSP	Easy		Hard		**STP**	Easy		Hard	
Method	Gap ↓	Time ↓	Gap ↓	Time ↓	Method	Gap ↓	Time ↓	Gap ↓	Time ↓
CPLEX	**0.00%**	702s	**0.01%**	3600s	SCIP-Jack	**0.00%**	22s	**0.50%**	717s
MPGN	12.78%	9s	1.50%	69s	RL	14.00%	31s	13.10%	1s
L-RHO	27.20%	21s	1.03%	58s	SL	14.00%	31s	13.10%	1s
FunSearch	5.05%	3600s	12.10%	3600s	FunSearch	8.29%	3600s	5.82%	3600s
Self-Refine	6.66%	3600s	1.14%	3600s	Self-Refine	11.23%	3600s	6.93%	3600s
ReEvo	5.61%	3600s	2.16%	3600s	ReEvo	14.36%	3600s	10.03%	3600s

C. Example Prompt

Our query prompts basically consist of two parts: the description of the problem background and the starter code for LLM to fill in. The following is an example prompt on TSP.

The evaluation example

Problem Description

The Traveling Salesman Problem (TSP) is a classic combinatorial optimization problem where, given a set of cities with known pairwise distances, the objective is to find the shortest possible tour that visits each city exactly once and returns to the starting city. More formally, given a complete graph G = (V, E) with vertices V representing cities and edges E with weights representing distances, we seek to find a Hamiltonian cycle (a closed path visiting each vertex exactly once) of minimum total weight.

Starter Code

```
def solve(**kwargs):
    """
    Solve a TSP instance.
```

```
    Args:
        - nodes (list): List of (x, y) coordinates representing cities in the
            TSP problem
                  Format: [(x1, y1), (x2, y2), ..., (xn, yn)]

    Returns:
        dict: Solution information with:
            - 'tour' (list): List of node indices representing the solution
                path
                        Format: [0, 3, 1, ...] where numbers are indices into
                            the nodes list
    """
    # Your function must yield multiple solutions over time, not just return
        one solution
    # Use Python's yield keyword repeatedly to produce a stream of solutions
    # Each yielded solution should be better than the previous one
    while True:
        yield {
            'tour': [],
        }
```