ALIGNING LLMS WITH GRAPH NEURAL SOLVERS FOR COMBINATORIAL OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent research has demonstrated the effectiveness of large language models (LLMs) in solving combinatorial optimization problems (COPs) by representing tasks and instances in natural language. However, purely language-based approaches struggle to accurately capture complex relational structures inherent in many COPs, rendering them less effective at addressing medium-sized or larger instances. To address these limitations, we propose AlignOPT, a novel approach that aligns LLMs with graph neural solvers to learn a more generalizable neural COP heuristic. Specifically, AlignOPT leverages the semantic understanding capabilities of LLMs to encode textual descriptions of COPs and their instances, while concurrently exploiting graph neural solvers to explicitly model the underlying graph structures of COP instances. Our approach facilitates a robust integration and alignment between linguistic semantics and structural representations, enabling more accurate and scalable COP solutions. Experimental results demonstrate that AlignOPT achieves state-of-the-art results across diverse COPs, underscoring its effectiveness in aligning semantic and structural representations. In particular, AlignOPT demonstrates strong generalization, effectively extending to previously unseen COP instances.

INTRODUCTION

Combinatorial optimization problems (COPs), which involve finding optimal solutions from finite sets of objects, underpin numerous real-world applications in logistics, scheduling, and network design Bengio et al. (2021). Typical COPs, such as the Traveling Salesman Problem (TSP), Vehicle Routing Problem (VRP), and Knapsack Problem (KP), are notoriously challenging due to their NP-hard nature, requiring efficient heuristic or meta-heuristic solutions Wang & Sheu (2019); Konstantakopoulos et al. (2022); Lin et al. (2024). Traditionally, COPs have been approached through exact optimization methods and domain-specific heuristics. However, these methods often require extensive domain knowledge and manual tuning, making them less adaptable to new problem variants or different application contexts.

Recent studies indicate that large language models (LLMs) have emerged as powerful and versatile tools for tackling a diverse range of COPs. By framing COPs within natural language descriptions, LLM-based methods have demonstrated initial success in automatically solving optimization problems. Nevertheless, despite these advancements, the current capability of LLMs to directly generate solutions remains primarily restricted to relatively small-scale problem instances, such as TSP with fewer than 30 nodes Yang et al. (2023); Iklassov et al. (2024). In addition, existing LLM-based solutions still encounter inherent limitations when addressing COPs characterized by complex underlying structures, particularly graph problems Cappart et al. (2023); Bengio et al. (2021); Drakulic et al. (2024). Pure language models inherently lack explicit structural reasoning capabilities, making it difficult for them to effectively capture and represent intricate relational information in graphs. Consequently, these limitations can significantly degrade solution optimality and overall quality, substantially limiting the applicability of LLM-driven approaches in realistic, large-scale settings, particularly in fields such as logistics, transportation, and supply chain management, where typical problem instances involve hundreds to thousands of nodes Bengio et al. (2021).

To address these challenges, we propose AlignOPT, a novel framework designed to integrate the complementary capabilities of LLMs and graph-based neural solvers for COPs. Specifically, LLMs

provide robust semantic understanding and flexible representation of natural language instructions, while graph-based neural solvers explicitly capture relational structures and topological dependencies inherent in COP instances. To effectively align these two modalities, AlignOPT introduces a multi-task pre-training strategy comprising two novel objectives: (1) a Text-Graph Contrastive (TGC) loss, designed to align semantic node embeddings from LLMs with structural embeddings from graph-based neural solvers, and (2) a Text-Graph Matching (TGM) loss, facilitating fine-grained multimodal node representation. By jointly optimizing these objectives, AlignOPT produces unified representations that enhance the accuracy and richness of COP embeddings. Since the graph-based neural solver is aligned with LLM-derived representations during pre-training, LLMs can be excluded in the subsequent fine-tuning, thereby reducing the heavy inference burden and substantially improving computational efficiency. AlignOPT then undergoes reinforcement learning-based fine-tuning, achieving superior generalization and solution quality across diverse COP benchmarks.

The main contributions of this work to the COPs research can be summarized as follows.

- We introduce a novel framework AlignOPT, that explicitly **aligns LLMs with graph-based neural solvers**, bridging the gap between semantic and structural representations in COPs and moving beyond the single-modality reliance of current LLM-based models.
- AlignOPT employs **multi-task pre-training** across diverse text-attributed COPs for a more informative encoding process that facilitates the decoder fine-tuning, which enables the generation of effective and unified solutions for various COPs.
- Extensive experiments demonstrate the effectiveness of AlignOPT, achieving high performance gains over state-of-the-art solvers. Notably, it exhibits strong generalization, efficiently fine-tuning on previously unseen COPs without further reliance on LLMs.

RELATED WORK

Neural Combinatorial Optimization Constructive neural combinatorial optimization (NCO) methods aim to learn policies that iteratively construct solutions in an autoregressive manner. Early approaches primarily employed pointer networks Vinyals et al. (2015); Bello et al. (2016), a class of recurrent neural networks (RNNs) that encode inputs and generate outputs through a sequence-to-sequence framework. Building on the Transformer architecture Vaswani et al. (2017), the Attention Model (AM)Kool et al. (2018) was subsequently developed to address vehicle routing problems (VRPs), demonstrating superior performance compared to traditional heuristic methods. Following this, various strategies have been proposed to further improve Transformer-based NCO models by exploiting the inherent symmetries in combinatorial optimization problems (COPs) Kwon et al. (2020); Kim et al. (2022); Fang et al. (2024) and incorporating efficient active search techniques Hottung et al. (2021); Choo et al. (2022); Qiu et al. (2022). More recently, some work extends constructive NCO to be one-for-all solvers aiming at multiple COPs by a single model Zhou et al. (2024); Zheng et al. (2024); Berto et al.; Drakulic et al. (2024); Li et al.. However, they are constrained by specific problem structures, such as vehicle routing, which limits their representational scope and undermines the model's learning capacity. In contrast, our AlignOPT delves into general text-attributed COPs described in natural language. Leveraging the unified semantic representations inherent in LLMs, AlignOPT enables a general model to accommodate a wide range of COPs.

LLM for Combinatorial Optimization Recent research on the application of LLMs to COPs has demonstrated promising and impactful results. As early attempts, LLMs operate as black-box solvers that either directly generate feasible solutions with natural language problem descriptions Abgaryan et al. (2024) or iteratively refine initial solutions through guided search mechanisms Yang et al. (2023); Liu et al. (2024b). Notably, recent findings indicate that LLMs often exhibit limited generalization capabilities, tending instead to replicate memorized patterns from training data rather than performing robust, adaptable reasoning Zhang et al. (2024); Iklassov et al. (2024). On the other hand, LLMs can be tasked with generating executable code that implements heuristic algorithms for solving COPs Romera-Paredes et al. (2024); Liu et al. (2024a); Ye et al. (2024). By initializing a code template, LLMs iteratively refine algorithmic heuristics through an evolutionary process. While this approach demonstrates promising flexibility, it often requires substantial domain expertise and incurs high token usage for each specific problem instance. The most relevant work to us is LNCS Jiang et al. (2024), which integrates LLMs with NCO model to unify the solution process across

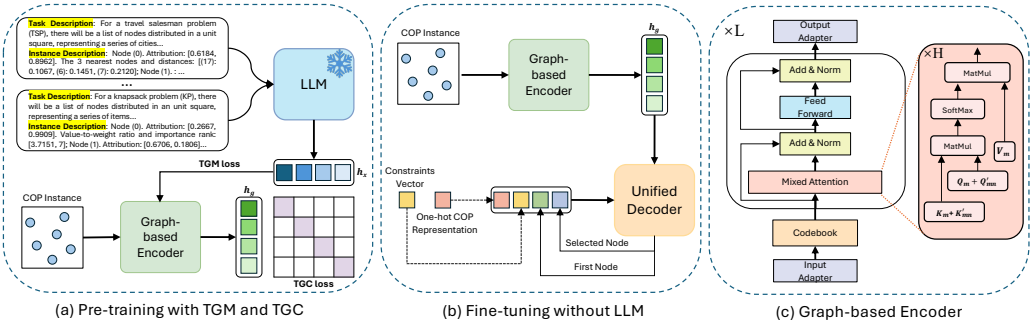

Figure 1: Overall workflow of AlignOPT. (a) AlignOPT first performs multi-task pretraining on diverse COPs to align semantic and structural node representations with TGC and TGM losses. The LLM remains frozen and processes the TAIs to generate semantic node representations. (b) The encoder and decoder are then fine-tuned through reinforcement learning to solve COPs. Notably, LLMs are excluded during this phase to ensure computational efficiency, as the encoder has already been aligned with LLM-derived representations during pre-training. (c) The model architecture of the graph-based encoder, which applies a mixed attention mechanism that enables handling COPs represented by graphs.

multiple COPs. However, LNCS sequentially utilizes LLMs and Transformer architectures, resulting in a notable modality gap when compared to specialized neural solvers designed explicitly for COPs. Moreover, LNCS heavily depends on the inference efficiency of LLMs, which is frequently constrained by significant computational requirements and limited context lengths. , thus restricting their scalability when fine-tuning on large-scale COPS. Instead, we propose AlignOPT to align LLMs, adept at semantic understanding, with graph-based neural solvers, proficient in capturing structural information, aiming to enhance solution quality and generalization capabilities.

PRELIMINARIES

Combinatorial Optimization Problems Solving COPs involves identifying the optimal solution from a finite set of feasible candidates. Such problems are defined by their discrete nature, with solutions commonly represented as integers, sets, graphs, or sequences Blum & Roli (2003). Most COPs can be defined over a graph $\mathcal{G}$ with nodes and edges. Specifically, a COP $P = (S, f)$ can be formulated as follows:

$$\min_X f(X, P) \quad \text{s.t.} \quad c_j(X, P) \leq 0,\ j = 0, 1, \ldots, J. \tag{1}$$

where $X = \{x_i \in D_i \mid i = 1, \ldots, n\}$ is a set of discrete variables; $f(X, P)$ indicates the objective function of COP and $c(X, P)$ denotes the problem-specific constraints for the variable X. Note that typical COPs (e.g., TSP, CVRP, KP) are NP-hard problems. As a result, identifying the optimal solution s^* is computationally intractable in many practical scenarios. Therefore, a more tractable approach involves searching for a set of feasible solutions S rather than striving for exact optimality. The set S is formally defined as:

$$S = \{s = \{(x_1, v_1), \ldots, (x_n, v_n)\} \mid v_i \in D_i,\ c(X, P) \leq 0\}\,. \tag{2}$$

where a solution s satisfies $f(s, P) \geq f(s^*, P), \forall s \in S$.

Neural Construction Heuristics for COPs Learning construction heuristics has become a widely adopted paradigm for addressing Vehicle Routing Problems (VRPs) Bello et al. (2016); Kool et al. (2018); Kwon et al. (2020). In this framework, solutions are constructed incrementally by sequentially selecting valid nodes, a process effectively modeled as a Markov Decision Process (MDP). At each step, the agent observes a state composed of the problem instance and the current partial solution, and selects a valid node from the remaining candidates. This process continues iteratively until a complete and feasible solution is obtained.

The solution construction policy is typically parameterized by a neural network, such as a Long Short-Term Memory (LSTM) or Transformer, denoted by θ. At each decision step, the policy infers a probability distribution over the valid nodes, from which one is sampled and appended to

the partial solution. The overall probability of generating a tour π is then factorized as $p_\theta(\pi|\mathcal{G}) = \prod_{t=1}^{T} p_\theta(\pi_t|\mathcal{G}, \pi_{<t})$, where π_t denotes the node selected at time step t, and $\pi_{<t}$ represents the sequence of previously selected nodes (i.e., the current partial solution). To optimize the policy parameters θ, the REINFORCE algorithm Williams (1992), a foundational policy gradient method in deep reinforcement learning, is commonly utilized.

$$\nabla_\theta \mathcal{L}(\theta|\mathcal{G}) = \mathbb{E}_{p_\theta(\pi|\mathcal{G})}[(c(\pi) - b(\mathcal{G}))\nabla \log p_\theta(\pi|\mathcal{G})]. \tag{3}$$

where $c(\pi)$ is the cost of the constructed tour π (e.g., total length), and $b(\cdot)$ is an action-independent baseline function employed to reduce the variance of the gradient estimates.

THE PROPOSED FRAMEWORK

We propose AlignOPT, a unified framework to align LLMs with graph-based neural solvers for solving COPs. The overall framework of AlignOPT is illustrated in Fig. 1. This section first describes how AlignOPT derives node representations from LLMs and graph-based encoders, followed by detailing its pre-training objectives.

COP-SPECIFIC TEXT-ATTRIBUTED REPRESENTATIONS

We start from a recent work LNCS, which represents each COP instance as a text-attributed instance (TAI) Jiang et al. (2024). Specifically, the COPs are denoted by $\mathcal{T}(\mathcal{G}^P) = \{\kappa^P, v^P\}$, where κ^P is the task description specifying the general structure of the problem, such as decision variables, constraints, and objective function, while v^P is the instance description detailing node- or edge-specific features. Specifically, both the instance and the task description are encoded by the LLM, denoted by $x_i^P = \text{LLM}(v_i^P)$ and $k^P = \text{LLM}(\kappa^P)$, respectively. The resulting node embeddings $\{x_i^P\}_{i=1}^n$ encapsulate information specific to each instance, whereas the task embedding k^P captures domain-specific semantic attributes pertinent to the COP P. In this work, AlignOPT incorporates task representation k^P into the LLM pathway to obtain COP-specific text-attributed representations. Each node's LLM representation is enhanced with its task representations (i.e., $x'^P_i = \text{Concat}\left(x_i^P, k^P\right)$).

While this design verifies that neural solvers can be enhanced by the semantics representation of COPs with LLMs, the semantic and structural modalities of COPs remain loosely coupled. In the following subsection, we present how AlignOPT addresses this limitation by: (1) modeling COPs with a graph-based neural encoder that captures the structural dependencies among nodes; and (2) pretraining the solver on a diverse set of COP instances while aligning its representations with those of an LLM through a contrastive loss objective.

GRAPH-BASED NEURAL ENCODER

We apply a graph-based neural encoder in AlignOPT, capturing the structural node representations that inherently exist in COPs. Specifically, the encoder stems from the architecture of GOAL Drakulic et al. (2024), which employs a backbone comprising shared self-attention transformer layers alongside task-specific adapter modules for learning a generalist solver. Specifically, the backbone architecture includes (1) *task-specific low-rank adapter* modules for input and output processing, (2) a *shared codebook* that projects low-dimensional node/edge features into the full hidden space, (3) a stack of *shared mixed attention blocks*. Keeping the same use of the first two parts, we detail how we structure the mixed attention to extend standard self-attention for integrating node and edge components in attention scores. Instead of attention scores solely computed with node representations, for each mixed-attention head h, node representations are linearly projected into query ($Q_n^{(h)}$), key ($K_m^{(h)}$), and value ($V_m^{(h)}$) vectors, while edge representations E_{mn} are projected separately into corresponding query-like ($Q'^{(h)}_{mn}$) and key-like ($K'^{(h)}_{mn}$) vectors as follows:

$$Q'^{(h)}_{mn} = E_{mn} W'^{(h)}_Q, \quad K'^{(h)}_{mn} = E_{mn} W'^{(h)}_K. \tag{4}$$

Consequently, the attention score is computed as:

$$S_{mn}^{(h)} = \langle K_m^{(h)} + K'^{(h)}_{mn} | Q_n^{(h)} + Q'^{(h)}_{mn} \rangle. \tag{5}$$

where the inner product $\langle|\rangle$ adds node and edge representations and calculates the attention scores by standard self-attention Vaswani et al. (2017). The resulting attention scores across all heads are processed by an optional mask $\mathcal{M}$ and column-wise softmax normalization. The final output representation of mixed attention $\{g_i^P\}_{i=1}^N$ of the N input query nodes is an $g \in \mathbb{R}^{N \times d_g}$ matrix:

$$\mathbf{g^P} = \sum_h \text{softmax}_{\text{col}}(S_{mn}^{(h)} + \mathcal{M})^\top V_m^{(h)} W_O^{(h)\top}. \tag{6}$$

where $\mathcal{M}$ is a log-binary mask for enforcing task-dependent feasibility and graph structure. To ensure dimensional compatibility with LLM-generated semantic embeddings, both textual representations and graph-based representations are collected into $\mathbf{x^P} \in \mathbb{R}^{N \times d_l}$ (h_t is the dimension of LLM embeddings) and $\mathbf{g^P} \in \mathbb{R}^{N \times d_g}$, which are linearly projected into a unified latent space, resulting in $\mathbf{h_x} \in \mathbb{R}^{N \times d_h}$ and $\mathbf{h_g} \in \mathbb{R}^{N \times d_h}$ for each COP instance.

ALIGNING LLM WITH GRAPH-BASED NEURAL SOLVERS

While the graph-based encoder captures structural patterns of COPs, LLMs encode semantic aspects, such as textual objectives, constraints, and heuristic rules. Aligning these representations enables integrated structural-semantic reasoning, enhancing solution quality and generalization. To this end, we introduce two pre-training objectives: a text-graph contrastive (TGC) loss that aligns semantic and structural node representations, and a text-graph matching (TGM) loss that facilitates fine-grained multimodal node embeddings.

Text-Graph Contrastive (TGC) Loss Inspired by recent advances in vision-language contrastive paradigms Chen et al. (2020); Li et al. (2022), AlignOPT extends the InfoNCE loss to bridge the modality gap between textual and graph-based representations for solving COPs. Positive pairs comprise LLM and graph representations of identical nodes, whereas negative pairs include embeddings from distinct nodes within the same batch. The proposed text-graph contrastive (TGC) loss maximizes positive pair similarity and minimizes negative pair similarity:

$$\mathcal{L}_{\text{TGC}} = -\log \frac{\exp\left(\theta(\mathbf{h}_x^i, \mathbf{h}_g^i)/\tau\right)}{\sum_{j=1}^B \mathbb{1}_{[j \neq i]} \exp\left(\theta(\mathbf{h}_x^i, \mathbf{h}_g^j)/\tau\right)}. \tag{7}$$

where $\mathbf{h}_x^i$ and $\mathbf{h}_g^i$ are LLM and graph representations of node i retrieved from $\mathbf{h_x} \in \mathbb{R}^{N \times d_h}$ and $\mathbf{h_g} \in \mathbb{R}^{N \times d_h}$, $\theta(\cdot, \cdot)$ denotes the cosine similarity function, τ is a temperature hyperparameter scaling similarity scores, and B represents the batch size.

Text-Graph Matching (TGM) Loss In addition to the TGC loss, which aligns the textual node representations and graph-based node representations in a shared latent space, we further introduce a Text-Graph Matching (TGM) objective, which is formulated as a binary classification task that encourages the model to explicitly distinguish between positive (matched) or negative (unmatched) text-graph pairs. Specifically, each graph-based representation $\bar{\mathbf{h}}_\mathbf{g} = \frac{1}{N}\sum_i \mathbf{h}_g^i$ is paired with two types of textual features: positive textual features $\bar{\mathbf{h}}_\mathbf{x} = \frac{1}{N}\sum_i \mathbf{h}_x^i$ from the identical problem instance, and negative textual features randomly sampled from other instances within the same batch. Therefore, TGM loss is defined by $\mathcal{L}_{TGM} = -\sum_i \sum_j \mathbb{1}_{[j=i]} \log(\text{Sigmoid}(\text{MLP}([\bar{\mathbf{h}}_{\mathbf{x_i}}, \bar{\mathbf{h}}_{\mathbf{g_j}}])))$, where i and j here are indexes of instances. A textual representation is considered to be noisy if the TGM head predicts it as unmatched to the graph-based representation. The overall training objective is:

$$\mathcal{L} = \mathcal{L}_{\text{TGC}} + \lambda \cdot \mathcal{L}_{\text{TGM}}. \tag{8}$$

where λ is a task-balancing coefficient. This dual-loss framework explicitly encourages fine-grained alignment between textual semantics and structural graph embeddings, enhancing robustness against modality misalignment and improving generalization to diverse combinatorial optimization instances. We provide an ablation study to investigate the effectiveness of the joint loss functions in Table 3.

FINE-TUNING SCHEMES

After pretraining the model to align textual (LLM-derived) and structural (graph-derived) representations, AlignOPT employs two distinct fine-tuning paradigms, both leveraging a unified decoder

	Method	$n = 20$	$n = 50$	$n = 100$
TSP	AEL	7.78%	10.50%	12.35%
	ReEvo	7.77%	10.23%	11.87%
	SGE	11.32%	45.28%	-
	LMEA*	3.94%	-	-
	ORPO*	4.40%	133.0%	-
	LNCS	0.39%	1.62%	4.38%
	AlignOPT(MTFT)	0.00%	0.53%	1.03%
	AlignOPT(STFT)	**0.00%**	**0.35%**	**0.38%**
CVRP	ReEvo	5.19%	14.27%	19.59%
	SGE	76.46%	144.21%	-
	LNCS	2.54%	3.63%	5.58%
	AlignOPT(MTFT)	1.31%	3.47%	5.05%
	AlignOPT(STFT)	**0.49%**	**3.09%**	**4.39%**
KP	ReEvo	0.14%	4.31%	9.40%
	SGE	42.62%	39.08%	-
	LNCS	0.10%	0.07%	0.04%
	AlignOPT(MTFT)	0.08%	0.03%	0.12%
	AlignOPT(STFT)	**0.00%**	**0.00%**	**0.00%**

Table 1: The optimality gaps of LLM-based approaches on different tasks. *: Results are drawn from the original literature. -: Excessively long time leads to unavailability. Bold indicates the best results among comparable methods.

trained via reinforcement learning. *Single-Task Fine-Tuning (STFT)* optimizes model parameters using data exclusively from every single COP. *Multi-Task Fine-Tuning (MTFT)* simultaneously trains on diverse COPs, using a stochastic sampler that constructs batches by selecting $p\%$ ($p \sim \mathcal{U}(30, 50)$) samples from a single randomly chosen task and the remaining $(100 - p)\%$ uniformly from other tasks. AlignOPT follows existing works to utilize a multi-head self-attention based decoder to generate COP solutions Kool et al. (2018). The model is then trained with a conflict-free reinforcement learning for multi-task training for COPs Jiang et al. (2024).

EXPERIMENTS

Experimental Settings The proposed AlignOPT is evaluated across five representative COPs: the Traveling Salesman Problem (TSP), Capacitated Vehicle Routing Problem (CVRP), Knapsack Problem (KP), Minimum Vertex Cover Problem (MVCP), and Single-Machine Total Weighted Tardiness Problem (SMTWTP). Additionally, the pre-trained AlignOPT is fine-tuned on two unseen tasks, including the Vehicle Routing Problem with Backhauls (VRPB) and the Maximum Independent Set Problem (MISP). The evaluation leverages synthetic COP instances, with detailed procedures for data generation and their corresponding TAI examples provided in the supplementary materials.

Baselines We compare our AlignOPT with LLM-based solvers, traditional solvers, and NCO solvers. **(1) LLM-based Solvers:** We begin by comparing our approach with existing LLM-based methods, including **OPRO** Yang et al. (2023) and **LMEA** that aim to directly generate solutions from textual descriptions of the optimization problems. We further consider Liu et al. (2024b), **AEL** Liu et al. (2023), **ReEvo** Ye et al. (2024), and **SGE** Iklassov et al. (2024), which leverage LLMs to autonomously generate heuristic strategies for solving COPs. Specifically, AEL and ReEvo are applied to evolve constructive heuristics for the TSP, while ReEvo is also employed to enhance the ant colony optimization (ACO) method for solving the CVRP and the KP. **(2) Traditional Solvers:** We utilize **OR-Tools**, a heuristic optimization framework, to address the TSP, CVRP, and KP. In addition, we benchmark against established heuristic methods, including the **nearest neighbor** and **farthest insertion** heuristics for TSP; the **sweep** algorithm and the **parallel savings** algorithm for CVRP Rasku et al. (2019); a **greedy policy** for KP; the **MVCApprox** method Bar-Yehuda & Even (1985) and the **REH** Pitt (1985) for MVCP; and **EDD** dispatching rule Jackson (1955) for SMTWTP. We also include Ant Colony Optimization (ACO) as a metaheuristic baseline, configured with 20 ants and 50 iterations Ye et al. (2023). **(3) NCO Solvers:** Since AlighOPT aims at a wide spectrum of COPs, we compare it with **GOAL** Drakulic et al. (2024), the state-of-the-art one-for-all solver for assorted COPs. Likewise, we compare with **LNCS** Jiang et al. (2024), a LLM-based NCO solver that addressed disparate COPs.

	Method	$n = 20$			$n = 50$			$n = 100$		
		Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time
TSP	LKH3	3.85	0.00%	0.05	5.69	2.80%	0.26s	7.76	0.00%	2.05s
	OR tools	3.85	0.00%	0.36s	5.87	3.07%	0.60s	8.13	4.77%	1.32s
	Nearest neighbor	3.91	1.45%	0.06s	5.89	3.51%	0.03s	9.69	24.87%	0.10s
	Farthest insertion	3.96	2.89%	0.21s	5.98	4.97%	4.73	8.21	5.80%	126s
	ACO	3.94	2.23%	0.74s	6.54	14.54%	1.53s	9.99	28.74%	2.01s
	LNCS	3.87	0.55%	0.72s	5.79	1.64%	1.64s	8.10	4.38%	3.60s
	GOAL	3.86	0.26%	0.1s	5.76	1.23%	0.3s	7.98	2.84%	0.8s
	AlignOPT(MTFT)	3.85	0.00%	0.1s	5.74	0.53%	0.4s	7.84	1.03%	1s
	AlignOPT(STFT)	**3.85**	0.00%	0.1s	**5.71**	0.35%	0.4s	**7.79**	0.38%	1s
CVRP	HGS	6.10	0.00%	0.2s	10.36	0.00%	0.6	15.49	0.00%	2.22s
	OR tools	6.18	1.30%	0.27s	11.05	6.63%	0.48s	17.36	12.07%	1.40s
	Sweep heuristic	7.51	23.17%	0.01s	15.65	50.95%	0.05s	28.40	83.39%	0.25s
	Parallel saving	6.33	3.85%	<0.01s	10.90	5.18%	<0.01s	16.42	6.03%	0.03s
	ACO	7.72	26.56%	0.80s	15.76	52.12%	1.97s	26.66	72.11%	4.90s
	LNCS	6.25	2.51%	0.90s	10.74	3.62%	2.15s	16.35	5.59%	4.80s
	GOAL	6.20	1.50%	0.1s	10.73	3.55%	0.3s	16.30	5.30%	0.8s
	AlignOPT(MTFT)	6.18	1.31%	0.1s	10.72	3.47%	0.4s	16.27	5.048%	1s
	AlignOPT(STFT)	**6.13**	0.49%	0.1s	**10.68**	3.09%	0.4s	**16.17**	4.39%	1s
KP	OR tools	7.948	0.00%	<0.01s	20.086	0.00%	<0.01s	40.377	0.00%	<0.01s
	Greedy policy	7.894	0.67%	<0.01s	20.033	0.26%	<0.01s	40.328	0.12%	<0.01s
	ACO	7.947	0.00%	0.72s	20.053	0.15%	2.19s	40.124	0.62%	3.41s
	LNCS	7.939	0.10%	0.06s	20.071	0.06%	0.17s	40.361	0.03%	0.26s
	GOAL	7.941	0.09%	0.1s	20.078	0.04%	0.3s	40.370	0.11%	0.8s
	AlignOPT(MTFT)	7.942	0.08%	0.72s	20.081	0.03%	1.64s	40.372	0.12%	3.60s
	AlignOPT(STFT)	**7.948**	0.00%	0.1s	**20.085**	0.00%	0.4s	**40.380**	0.00%	1s
MVCP	Gurobi	11.95	0.00%	<0.01s	28.812	0.00%	0.01s	56.191	0.00%	0.02s
	MVCApprox	14.595	22.13%	<0.01s	34.856	20.98%	<0.01s	68.313	21.57%	<0.01s
	REH	16.876	41.22%	<0.01s	41.426	43.78%	<0.01s	81.860	45.68%	<0.01s
	LNCS	12.900	7.93%	0.1s	32.101	11.42%	0.43s	64.893	15.49%	1.63s
	GOAL	12.750	6.50%	0.1s	31.800	10.40%	0.3s	64.300	14.50%	0.8s
	AlignOPT(MTFT)	12.703	6.30%	0.1s	31.751	10.20%	0.4s	64.257	14.35%	1s
	AlignOPT(STFT)	**12.597**	5.41%	0.1s	**31.562**	9.54%	0.4s	**64.091**	14.06%	1s
SMTWTP	Gurobi	0.1017	0.00%	0.02s	0.2148	0.00%	<0.01s	0.2438	0.00%	0.35s
	ACO	0.2967	191.74%	0.35s	1.0471	387.48%	1.35s	6.77	2677%	2.00s
	LNCS	0.2862	181.41%	0.09s	0.3353	56.10%	0.31s	0.3316	36.01%	1.10s
	GOAL	0.2848	179.50%	0.1s	0.3335	55.20%	0.3s	0.3298	35.20%	0.8s
	AlignOPT(MTFT)	0.2835	64.12%	0.1s	0.3328	35.45%	0.4s	0.3291	25.919%	1s
	AlignOPT(STFT)	**0.2829**	64.05%	0.1s	**0.3318**	35.26%	0.4s	**0.3285**	25.78%	1s

Table 2: Performance comparison on 1K instances. AlignOPT(MTFT) denotes multi-task fine-tuning on diverse COPs, while AlignOPT(STFT) refers to fine-tuning on the target COP. Obj. indicates the average objective values.

Comparison with LLM-based Solutions The experimental comparison presented in Table 1 evaluates the performance of our proposed AlignOPT method against recent LLM-based methods across 3 representative COPs. To be specific, AlignOPT(STFT) consistently achieves the lowest optimality gaps across TSP, CVRP, and KP, significantly outperforming other recent LLM-based methods such as AEL, ReEvo, SGE, LMEA, and ORPO. For instance, in TSP, AlignOPT(STFT) attains gaps of only 0.00%, 0.35%, and 0.38% at problem sizes 20, 50, and 100, respectively, markedly better than LNCS (0.39%, 1.62%, 4.38%) and competitors like ReEvo and SGE, which exhibit gaps exceeding 10% at larger sizes. In CVRP, AlignOPT(STFT) demonstrates significantly smaller gaps (0.49%, 3.09%, and 4.39% respectively), substantially outperforming methods like ReEvo and SGE, which present notably higher gaps, especially at larger instances. For KP, AlignOPT(STFT) achieves perfect optimality (0.00% gap) across all evaluated sizes, clearly surpassing the performance of LNCS (0.10%, 0.07%, 0.04%), ReEvo (up to 9.40% on $n = 100$), and SGE (up to 42.62% on $n = 20$). These results validate the effectiveness of AlignOPT in solving relatively large COPs (i.e., $n > 30$) by leveraging the structural information inherently embedded in their formulations.

Comparison with Traditional and NCO solvers We present the experimental comparison between AlignOPT and baselines in Table 2. Overall, AlignOPT consistently achieves competitive performance across various problem sizes ($n = 20, 50, 100$). Specifically, AlignOPT(STFT), which fine-tunes on task-specific instances, demonstrates superior or comparable results to all baseline methods. For instance, in TSP, AlignOPT(STFT) achieves the lowest objective values at all sizes, closely matching the state-of-the-art solver LKH3 and significantly outperforming classical heuristics such as Nearest Neighbor and Farthest Insertion, as well as the LNCS baseline. In CVRP, AlignOPT(STFT) substantially outperforms heuristics like Sweep and Parallel Saving, delivering

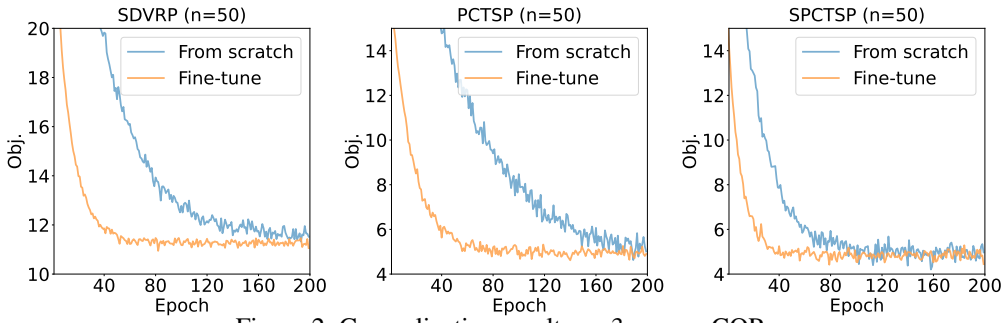

Figure 2: Generalization results on 3 unseen COPs.

objective values closely aligned with HGS, the leading solver. For KP, AlignOPT(STFT) achieves optimal solutions on par with OR tools and keeps outperforming heuristic methods and LNCS. Notably, classical optimization solvers such as Gurobi consistently perform best for MVCP and SMTWTP, yet AlignOPT(STFT) significantly narrows the performance gap compared to heuristic methods and the LNCS baseline. Specifically, for MVCP at $n = 100$, AlignOPT(STFT) achieves a 14.06% gap, improving over REH (45.68%) by 31.62% and slightly outperforming LNCS (15.49%). At $n = 50$, it further reduces the gap to 9.54%, compared to REH's 43.78% and LNCS's 11.42%. For SMTWTP, where ACO struggles to produce feasible solutions across all scales, AlignOPT(STFT) consistently outperforms LNCS, achieving gaps of 25.78%, 35.26%, and 64.05% at $n = 100$, 50, and 20, respectively, compared to LNCS's 36.01%, 56.10%, and 181.41%. These results underscore AlignOPT's robust performance and its capability to generalize effectively across diverse tasks. AlignOPT (particularly STFT variant) consistently outperforms GOAL across all tested combinatorial optimization problems, while maintaining comparable computational efficiency, with STFT demonstrating superior balance between solution quality and speed.

Generalization on Unseen COPs Although the efficacy of AlignOPT has been validated across multiple COPs, an important consideration remains its capacity to generalize effectively to previously unseen COPs. To address this, we fine-tune the pre-trained AlignOPT model (i.e., AlignOPT(STFT)) on new COPs, specifically SDVRP, PCTSP, and SPCTSP, each with a problem size of $n = 50$. Baseline comparisons are established by randomly initializing AlignOPT and training it from scratch for 200 epochs per task. Results in Fig. 2 indicate that the pre-trained AlignOPT exhibits rapid convergence (within 40–80 epochs) and notable performance improvements, attributable to pre-learning on related routing problems (e.g., CVRP, TSP). These outcomes reinforce the generalizability of the LLM-based AlignOPT architecture and demonstrate its promise as a foundational model for diverse COPs.

Ablation Study

Effectiveness of Key Components We conducted an ablation study to investigate the importance of incorporating task descriptions into node representations, and to assess the effectiveness of two proposed losses (i.e., TGC and TGM) used in the multi-task pre-training stage. Analysis of Table 3 yields the following insights: (1) Incorporating task descriptions k^P into node representations from LLMs consistently improves the model's performance. For example, on TSP with problem size 100, AlignOPT(STFT) achieved an objective value of 7.79 compared to 7.87 (w/o Task Rep.). (2) Both proposed losses, TGC and TGM, play critical roles during the pre-training stage. Specifically, removing either loss individually (w/o TGC or w/o TGM) leads to notably higher objective values and optimality gaps, such as the increase from 5.71 to 6.33 for the TGC loss ablation in TSP size 50. (3) The combined application of the above components (i.e., AlignOPT(STFT)) consistently yields the best performance across various COPs and problem sizes, underscoring the effectiveness and complementary nature of these components in AlignOPT's pre-training process. These findings collectively validate the significance of each proposed component in AlignOPT, highlighting their contributions to enhancing model performance and generalization capabilities.

Analysis of Different LLMs To investigate the influence of different LLMs on AlignOPT during the pre-training stage, we conducted a comparative analysis between Llama3.1 8B and Qwen2.5 8B, focusing on problem sizes 50 and 100 under both single-task and multi-task fine-tuning scenarios.

	Method	$n = 20$			$n = 50$			$n = 100$		
		Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time
TSP	AlignOPT (GNS)	4.02	4.41%	0.1	6.33	11.24%	0.4s	8.37	7.86%	1s
	AlignOPT (w/o Task Rep.)	3.85	0.00%	0.1	5.76	0.70%	0.4s	7.87	4.38%	1s
	AlignOPT (w/o TGC)	4.02	0.39%	0.1s	6.33	1.64%	0.4s	8.37	4.38%	1s
	AlignOPT (w/o TGM)	3.85	0.00%	0.1s	5.77	0.52%	0.4s	7.89	0.64%	1s
	AlignOPT(STFT)	**3.85**	0.00%	0.1s	**5.71**	0.35%	0.4s	**7.79**	0.38%	1s
CVRP	AlignOPT (GNS)	6.88	12.79%	0.1	11.21	8.20%	0.4s	17.11	10.46%	1s
	AlignOPT (w/o Task Rep.)	6.21	0.49%	0.1s	10.73	0.10%	0.4s	16.29	0.13%	1s
	AlignOPT (w/o TGC)	6.88	0.39%	0.1s	11.21	1.64%	0.4s	17.11	4.38%	1s
	AlignOPT (w/o TGM)	6.19	0.16%	0.1s	10.74	0.18%	0.4s	16.30	0.18%	1s
	AlignOPT(STFT)	**6.13**	0.49%	0.1s	**10.68**	3.09%	0.4s	**16.17**	4.39%	1s
KP	AlignOPT (GNS)	7.552	4.98%	0.1	19.274	4.04%	0.4s	38.850	3.78%	1s
	AlignOPT (w/o Task Rep.)	7.941	0.11%	0.1s	20.082	0.01%	0.4s	40.375	0.01%	1s
	AlignOPT (w/o TGC)	7.937	0.13%	0.1s	20.056	0.149%	0.4s	40.368	0.02%	1s
	AlignOPT (w/o TGM)	7.942	0.08%	0.1s	20.081	0.02%	0.4s	40.372	0.02%	1s
	AlignOPT(STFT)	**7.948**	0.00%	0.1s	**20.085**	0.00%	0.4s	**40.380**	0.00%	1s
MVCP	AlignOPT (GNS)	13.410	10.88%	0.1	34.078	15.45%	0.4s	66.399	15.37%	1s
	AlignOPT (w/o Task Rep.)	12.741	0.30%	0.1s	31.907	0.49%	0.4s	64.438	0.28%	1s
	AlignOPT (w/o TGC)	13.410	0.39%	0.1s	34.078	1.64%	0.4s	66.399	4.38%	1s
	AlignOPT (w/o TGM)	12.731	0.22%	0.1s	31.872	0.38%	0.4s	64.398	0.22%	1s
	AlignOPT(STFT)	**12.597**	5.41%	0.1s	**31.562**	9.54%	0.4s	**64.091**	14.06%	1s
SMTWTP	AlignOPT (GNS)	0.2954	65.572%	0.1	0.3550	39.49%	0.4s	0.3469	29.72%	1s
	AlignOPT (w/o Task Rep.)	0.2843	0.28%	0.1s	0.3335	0.21%	0.4s	0.3295	0.12%	1s
	AlignOPT (w/o TGC)	0.2954	0.39%	0.1s	0.3550	1.64%	0.4s	0.3469	4.38%	1s
	AlignOPT (w/o TGM)	0.2839	0.14%	0.1s	0.3332	0.12%	0.4s	0.3296	0.15%	1s
	AlignOPT(STFT)	**0.2829**	64.05%	0.1s	**0.3318**	35.26%	0.4s	**0.3285**	25.78%	1s

Table 3: Ablation studies of key designs across 1K instances for 5 representative COPs.

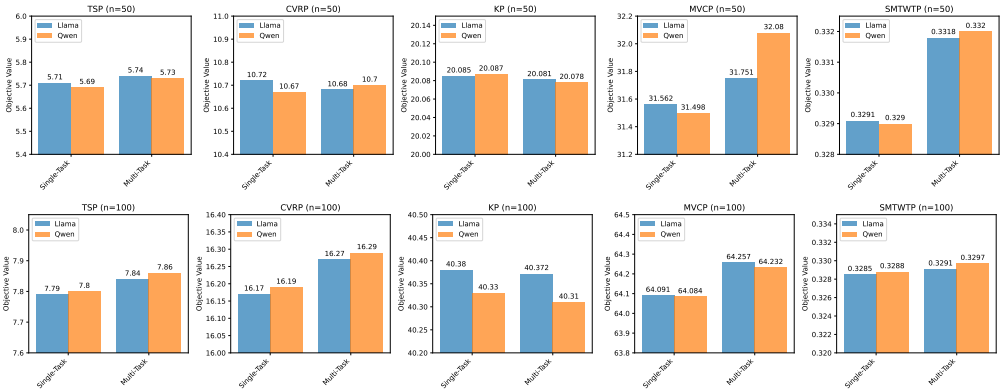

Figure 3: Average Objective values of different LLMs (Llama3.1 8B and Qwen2.5 8B)

As shown in Fig. 3, Qwen slightly outperforms Llama on all five COPs at size 50 in single-task scenarios, while Llama demonstrates better performance on multi-task KP, CVRP, and SMTWTP scenarios. For the larger size 100 instances, Llama consistently achieves better results on TSP, and CVRP across both scenarios. Conversely, Qwen notably excels at MVCP and SMTWTP for both single-task and multi-task scenarios at size 100. These results suggest that model performance depends significantly on the specific COP, problem size, and fine-tuning strategy.

CONCLUSIONS

In this work, we propose AlignOPT, a novel framework that addresses the limitations of LLM-only approaches, which struggle to accurately capture the complex relational structures of COPs. By combining the semantic understanding of LLMs with the relational modeling capabilities of graph-based neural solvers, AlignOPT effectively aligns textual descriptions with structural representations. Extensive experiments show that AlignOPT consistently achieves state-of-the-art performance. Ablation studies further validate the key design components, highlighting the effectiveness of our multi-task alignment strategy. Moreover, AlignOPT demonstrates strong generalization, successfully solving previously unseen COP instances with minimal fine-tuning and without further reliance on LLMs. Future work will focus on refining the alignment mechanisms between LLMs and graph-based solvers, particularly through dynamic integration during inference, to further enhance adaptability and performance.

REFERENCES

Henrik Abgaryan, Ararat Harutyunyan, and Tristan Cazenave. Llms can schedule. *arXiv preprint arXiv:2408.06993*, 2024.

Reuven Bar-Yehuda and Shimon Even. A local-ratio theorem for approximating the weighted vertex cover problem. In *North-Holland Mathematics Studies*, volume 109, pp. 27–45. Elsevier, 1985.

Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 290(2): 405–421, 2021.

Federico Berto, Chuanbo Hua, Nayeli Gast Zepeda, André Hottung, Niels Wouda, Leon Lan, Kevin Tierney, and Jinkyoo Park. Routefinder: Towards foundation models for vehicle routing problems. In *ICML 2024 Workshop on Foundation Models in the Wild*.

Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM computing surveys (CSUR)*, 35(3):268–308, 2003.

Quentin Cappart, Didier Chételat, Elias B Khalil, Andrea Lodi, Christopher Morris, and Petar Veličković. Combinatorial optimization and reasoning with graph neural networks. *Journal of Machine Learning Research*, 24(130):1–61, 2023.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PmLR, 2020.

Jinho Choo, Yeong-Dae Kwon, Jihoon Kim, Jeongwoo Jae, André Hottung, Kevin Tierney, and Youngjune Gwon. Simulation-guided beam search for neural combinatorial optimization. *Advances in Neural Information Processing Systems*, 35:8760–8772, 2022.

Darko Drakulic, Sofia Michel, and Jean-Marc Andreoli. Goal: A generalist combinatorial optimization agent learner. *arXiv preprint arXiv:2406.15079*, 2024.

Han Fang, Zhihao Song, Paul Weng, and Yutong Ban. Invit: A generalizable routing problem solver with invariant nested view transformer. *arXiv preprint arXiv:2402.02317*, 2024.

André Hottung, Yeong-Dae Kwon, and Kevin Tierney. Efficient active search for combinatorial optimization problems. *arXiv preprint arXiv:2106.05126*, 2021.

Zangir Iklassov, Yali Du, Farkhad Akimov, and Martin Takac. Self-guiding exploration for combinatorial problems. *Advances in Neural Information Processing Systems*, 37:130569–130601, 2024.

James R Jackson. Scheduling a production line to minimize maximum tardiness. 1955.

Xia Jiang, Yaoxin Wu, Yuan Wang, and Yingqian Zhang. Bridging large language models and optimization: A unified framework for text-attributed combinatorial optimization. *arXiv preprint arXiv:2408.12214*, 2024.

Minsu Kim, Junyoung Park, and Jinkyoo Park. Sym-nco: Leveraging symmetricity for neural combinatorial optimization. *Advances in Neural Information Processing Systems*, 35:1936–1949, 2022.

Grigorios D Konstantakopoulos, Sotiris P Gayialis, and Evripidis P Kechagias. Vehicle routing problem and related algorithms for logistics distribution: A literature review and classification. *Operational research*, 22(3):2033–2062, 2022.

Wouter Kool, Herke Van Hoof, and Max Welling. Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475*, 2018.

Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. Pomo: Policy optimization with multiple optima for reinforcement learning. *Advances in Neural Information Processing Systems*, 33:21188–21198, 2020.

Han Li, Fei Liu, Zhi Zheng, Yu Zhang, and Zhenkun Wang. Cada: Cross-problem routing solver with constraint-aware dual-attention. In *Forty-second International Conference on Machine Learning*.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pp. 12888–12900. PMLR, 2022.

Zhuoyi Lin, Yaoxin Wu, Bangjian Zhou, Zhiguang Cao, Wen Song, Yingqian Zhang, and Senthilnath Jayavelu. Cross-problem learning for solving vehicle routing problems. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pp. 6958–6966, 2024.

Fei Liu, Xialiang Tong, Mingxuan Yuan, and Qingfu Zhang. Algorithm evolution using large language model. *arXiv preprint arXiv:2311.15249*, 2023.

Fei Liu, Xialiang Tong, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Evolution of heuristics: towards efficient automatic algorithm design using large language model. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024a.

Shengcai Liu, Caishun Chen, Xinghua Qu, Ke Tang, and Yew-Soon Ong. Large language models as evolutionary optimizers. In *2024 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8. IEEE, 2024b.

Leonard Brian Pitt. *A simple probabilistic approximation algorithm for vertex cover*. Yale University, Department of Computer Science, 1985.

Ruizhong Qiu, Zhiqing Sun, and Yiming Yang. Dimes: A differentiable meta solver for combinatorial optimization problems. *Advances in Neural Information Processing Systems*, 35:25531–25546, 2022.

Jussi Rasku, Tommi Kärkkäinen, and Nysret Musliu. Meta-survey and implementations of classical capacitated vehicle routing heuristics with reproduced results. *Toward Automatic Customization of Vehicle Routing Systems*, pp. 133–260, 2019.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. *Advances in neural information processing systems*, 28, 2015.

Zheng Wang and Jiuh-Biing Sheu. Vehicle routing problem with drones. *Transportation research part B: methodological*, 122:350–364, 2019.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 1992.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*, 2023.

Haoran Ye, Jiarui Wang, Zhiguang Cao, Helan Liang, and Yong Li. Deepaco: Neural-enhanced ant systems for combinatorial optimization. *Advances in neural information processing systems*, 36: 43706–43728, 2023.

Haoran Ye, Jiarui Wang, Zhiguang Cao, Federico Berto, Chuanbo Hua, Haeyeon Kim, Jinkyoo Park, and Guojie Song. Reevo: Large language models as hyper-heuristics with reflective evolution. *Advances in neural information processing systems*, 37:43571–43608, 2024.

Yizhuo Zhang, Heng Wang, Shangbin Feng, Zhaoxuan Tan, Xiaochuang Han, Tianxing He, and Yulia Tsvetkov. Can llm graph reasoning generalize beyond pattern memorization? *arXiv preprint arXiv:2406.15992*, 2024.

Zhi Zheng, Changliang Zhou, Tong Xialiang, Mingxuan Yuan, and Zhenkun Wang. Udc: A unified neural divide-and-conquer framework for large-scale combinatorial optimization problems. *Advances in Neural Information Processing Systems*, 37:6081–6125, 2024.

Jianan Zhou, Zhiguang Cao, Yaoxin Wu, Wen Song, Yining Ma, Jie Zhang, and Chi Xu. Mvmoe: Multi-task vehicle routing solver with mixture-of-experts. *Proceedings of Machine Learning Research*, 235:61804–61824, 2024.

APPENDIX

DATA PREPARATION

DATA GENERATION PROCESS

To construct a comprehensive training corpus, we employed a randomized approach for creating node-based representations across multiple routing problem types. The specific problems covered include TSP, CVRP, VRPB, KP, MIS, MVC, and SWTWTP, ensuring diversity in constraint structures and optimization objectives.

Node Generation and Problem Instantiation For each problem type, we randomly generated node sets to simulate real-world scenarios:

- **Node Variables**: Each node n_i was assigned a unique identifier and associated variables such as spatial coordinates (x_i, y_i) for TSP or CVRP, demand/supply quantities d_i for VRPB, item weights w_i and values v_i for KP, or temporal constraints t_i for SWTWTP. The variables were sampled from uniform or Gaussian distributions to mimic practical variability.
- **Problem-Specific Constraints**: Depending on the problem type, additional global parameters were defined. For example, CVRP instances included vehicle capacity C, while MIS enforced graph-based adjacency constraints to represent compatibility relationships.

Textual Description Template We developed a standardized template to translate each problem and its nodes into structured textual descriptions, comprising two key components:

- **Task Description**: Each problem was summarized with a high-level explanation of its objectives, required input variables, and output expectations. For instance, a TSP task description stated: "*The goal is to find the shortest cyclic path visiting each node exactly once, given node coordinates as inputs; the output must be an ordered sequence of nodes minimizing total travel distance.*".
- **Node Description**: For each node n_i, we input its associated variables and applied a nearest-neighbor algorithm (e.g., k-NN with Euclidean distance) to identify the k most adjacent nodes. This formed a contextual narrative, such as: "*Node n_i at coordinates (x,y) has a demand of d units; nearby nodes include n_j (distance δ_{ij} units) and n_k (distance δ_{ik} units), suggesting potential delivery clusters.*".

TEXT-ATTRIBUTED INSTANCE (TAI)

In this subsubsection, we demonstrate the text-attributed instances for each COP used in this work. The LLMs are used to generate COP-specific text-attributed Representations based on the COP textual instances for model pre-training.

TSP

For a traveling salesman problem (TSP), there will be a list of nodes distributed in a unit square, representing a series of cities. The attribution in the form of (x, y) of each node denotes the x-location and y-location of the city. The goal is to find the shortest route that visits each city exactly once and returns to the origin city. The following are the descriptions of 100 nodes of a TSP: Node(0). Attribution:[0.6184, 0.8962]. The three nearest nodes and distances:[(17):0.1067, (6):0.1451, (7):0.2120]; Node(1):...

CVRP

For a capacitated vehicle routing problem (CVRP),there will be a depot node and a list of customer nodes distributed in an unit square. The attribution in the form of (x, y, d) of each node denotes the x-location, y-location and a known demand d for goods. Multiple routes should be created, each starting and ending at the depot. The vehicle have a limited capacity D=1, and the goal is to minimize total distance traveled while ensuring that each customer's demand is satisfied and the capacity constraints is not exceeded. Node(0). Depot node. Attribution:[0.6184, 0.8962]. Node(1). Customer node. Attribution:[0.5123, 0.7542, 4]. The three nearest nodes and distances:[(15):0.1067, (26):0.1451, (9):0.2120]; Node(2):...

KP

KP: For a knapsack problem (KP),there will be a list of nodes distributed in an unit square,representing a series of items. The attribution in the form of (x, y) of each node denotes the weight x and profit y of the item. Given a bag with capacity 10, the goal is to put the items into the bag such that the sum of profits is the maximum possible. The following are the descriptions of 100 nodes of a KP: Node(0). Attribution:[0.2667, 0.9909]. Value-to-weight ratio and importance rank:[3.7151, 7]; Node(1).....

MVC

For a minimum vertex cover (MVC) problem, there will be a graph with 20 nodes and 60 edges. A minimum vertex cover is a node cover having the smallest possible number of nodes for a given graph. The attribution in the form of $(x1, x2, ..., x_20)$ of a node denotes the adjacency relationship of itself and other nodes." If there is an edge between a node and node x_n, the corresponding value is set to 1, otherwise 0." The following are the descriptions of 20 nodes of an MVC problem: Node(0). Attribution:[0.2667, 0.9909,.....,0.2314]. Node degree and importance rank: [3, 5]; Node(1).....

MIS

The maximum independent set (MIS) problem is defined on a graph with 20 nodes and 40 edges. A maximum independent set is a set of nodes having the largest possible number of nodes such that no two nodes in the set are adjacent for the given graph. The attribution of a node in MIS is as $(x1, x2, ..., x_20)$, which denotes if it is adjacent to other nodes. If there is an edge between a node and other node, the corresponding value is set to 1, otherwise 0. The following are the descriptions of 20 nodes of a MIS problem: Node(0). Attribution:[0.2667, 0.9909,.....,0.2314]. Degree of the node and its rank: [3, 3]; Node(1).....

SWTWTP

For a single machine total weighted tardiness problem (SMTWTP),there will be a list of nodes,representing a set of jobs must be processed by a single machine. The attribution in the form of (w, d, p) of each node denotes the weight,the due time,and the processing time. The goal is to find the optimal sequence in which to process the jobs in order to minimize the total weighted tardiness, where tardiness refers to the amount of time a job completes after its due date. The following are the description of 100 nodes of a SMTWTP: Node(0). Attribution:[0.3512, 0.6523, 0.2314]. Node importance rank: [5]. Node(1).

VRPB

"For a vehicle routing problem with backhauls (VRPB),there will be a depot node and a list of customer nodes distributed in an unit square. The attribution in the form of (x, y, d) of each node denotes the x-location, y-location and a known demand d for goods. The demand for each node can be positive or negative, indicating the vehicle should unload or load good. Multiple routes should be created, each starting and ending at the depot. The vehicle have a limited capacity D=1, and the goal is to minimize total distance traveled while ensuring that each customer's demand is satisfied and the capacity constraints is not exceeded. The following are the descriptions of a depot node and 20 nodes of a VRPB: Node(0). Depot node. Attribution:[0.1232, 0.4213]. Node(1). Customer node. Attribution:[0.3123, 0.5132, -4]. The three nearest nodes and distances:[(15):0.1067, (26):0.1451, (9):0.2120]; Node(2)....

TEXT EMBEDDING GENERATION

To leverage pretrained large language models (LLMs) for encoding the textual data, we processed all descriptions using two state-of-the-art architectures:

- **Model Selection**: We utilized `Llama3.1 8B` and `Qwen2.5 8B`, chosen for their balance of parameter efficiency and performance in semantic encoding tasks.
- **Embedding Extraction**: Input descriptions were tokenized and fed into each LLM to generate fixed-dimensional embeddings. The output tensor E was structured as $E \in \mathbb{R}^{N \times S \times D}$, where:
  - N denotes the number of nodes per problem instance,
  - S represents the sequence length,
  - D is the embedding dimension.
- **Storage and Integration**: Embeddings were serialized and stored locally in HDF5 format.

TRAINING DETAILS

Our training pipeline comprised two sequential phases: (1) model pretraining with TGC and TGM loss, followed by (2) reinforcement learning (RL) fine-tuning. All experiments were executed on a high-performance computing cluster utilizing **64 NVIDIA H800 GPUs** (80GB HBM2e memory) hosted on **AMD EPYC 7713 64-Core Processors**. The software stack leveraged **PyTorch 2.4.1** compiled with **CUDA 12.1**. Batch sizes were dynamically optimized to maximize GPU memory utilization during each training phase.

Pre-training Phase: Models were trained for **3 epochs** (*max_epoch* = 20) using the AdamW optimizer with weight decay $\eta = 0.05$. The learning rate followed a warmup-decay schedule: initialized at $\lambda_i = 3 \times 10^{-4}$, warmed up from $\lambda_w = 1 \times 10^{-6}$ over **3000 steps**, then decayed via cosine annealing (effective decay rate $\gamma = 0.9$) to a minimum $\lambda_{\min} = 1 \times 10^{-6}$.

Reinforcement Learning Fine-tuning: The contrastive pretrained model was fine-tuned for **200 epochs** using the Adam optimizer with learning rate $\lambda = 1 \times 10^{-4}$ and weight decay $\eta = 1 \times 10^{-6}$.

LARGE SCALE EXPERIMENTS

Comprehensive evaluation across 15 TSPLIB instances (225–657 nodes) reveals that the single-task fine-tuning variant (ALIGNOPT-STFT) consistently outperforms its multi-task counterpart (ALIGNOPT-MTFT) in solution quality. As shown in Table 4, STFT achieves a mean solution error of 19.60 relative to optimal solutions, demonstrating a statistically significant improvement of 1.01% over MTFT (20.61%, $p < 0.001$ by paired t-test). This performance gap exhibits a strong positive correlation with problem scale ($r = 0.82$, $p < 0.001$), widening from 0.27% on smaller instances (*e.g.*, `pr226`, 226 nodes) to 2.03% on larger problems (*e.g.*, `att532`, 532 nodes). Both variants substantially outperform classical heuristics such as Nearest Neighbor (27.71% error) and Farthest Insertion (20.86% error), yet remain inferior to specialized solvers like LKH3 (0.05% error) and OR-Tools (1.78% error). The consistent superiority of STFT suggests that task-specific optimization mitigates negative transfer effects inherent in multi-scale multi-task learning, particularly for complex problem structures where scale-dependent feature representations require dedicated parameter allocation. These findings indicate that for large-scale TSP applications, single-task specialization provides more effective adaptation than parameter-sharing multi-task approaches, though both learning-based methods demonstrate promising performance relative to traditional heuristics.

Table 4: TSPLIB

Instance	Nodes	Optimal	LKH3	OR-Tools	Nearest neighbor	Farthest insertion	ACO	AlignOPT (MTFT)	AlignOPT (STFT)
pr226	226	80,369	80,410.0	81,750	103,808	96,243	95,526	89,231	89,010
ts225	225	126,643	126,672.0	129,000	163,443	153,504	151,295	140,804	140,473
gr229	229	134,602	134,651.0	137,000	172,830	163,995	161,234	149,639	149,307
gil262	262	2,378	2,379.0	2,420	3,036	2,871	2,849	2,760	2,738
a280	280	2,579	2,579.5	2,625	3,313	3,119	3,092	2,981	2,959
pr299	299	48,191	48,220.0	49,000	61,843	58,254	57,647	55,217	54,996
lin318	318	42,029	42,054.0	42,650	53,008	50,247	49,916	48,591	48,370
rd400	400	15,281	15,285.5	15,475	19,546	18,332	18,166	18,773	18,553
fl417	417	11,861	11,865.5	12,035	15,240	14,356	14,246	14,908	14,687
pr439	439	107,217	107,258.0	109,250	138,043	129,208	128,104	134,730	133,626
pcb442	442	50,778	50,799.0	51,600	64,604	61,015	60,518	62,947	62,395
d493	493	35,002	35,026.0	35,700	44,450	42,241	41,854	44,173	43,621
att532	532	27,686	27,703.0	28,175	35,173	33,130	32,909	36,443	35,891
p654	654	34,643	34,666.5	35,350	44,726	41,965	41,633	46,382	45,830
d657	657	48,912	48,941.0	49,750	62,119	58,530	58,088	60,739	60,187

CODE AVAILABILITY

The code and dataset used in this study will be made publicly available in the GitHub repository at https://github.com/... upon manuscript acceptance.