GLOP: Learning Global Partition and Local Construction for Solving Large-Scale Routing Problems in Real-Time

Haoran Ye[1], **Jiarui Wang**[1], **Helan Liang**[1*], **Zhiguang Cao**[2], **Yong Li**[3], **Fanzhang Li**[1]
[1]Soochow University, China
[2]Singapore Management University, Singapore
[3]Tsinghua University, China
{hrye,jrwangfurffico}@stu.suda.edu.cn, {hlliang,lfzh}@suda.edu.cn
zgcao@smu.edu.sg, liyong07@tsinghua.edu.cn

Abstract

The recent end-to-end neural solvers have shown promise for small-scale routing problems but suffered from limited real-time scaling-up performance. This paper proposes GLOP (**G**lobal and **L**ocal **O**ptimization **P**olicies), a unified hierarchical framework that efficiently scales toward large-scale routing problems. GLOP partitions large routing problems into Travelling Salesman Problems (TSPs) and TSPs into Shortest Hamiltonian Path Problems. For the first time, we hybridize non-autoregressive neural heuristics for coarse-grained problem partitions and autoregressive neural heuristics for fine-grained route constructions, leveraging the scalability of the former and the meticulousness of the latter. Experimental results show that GLOP achieves competitive and state-of-the-art real-time performance on large-scale routing problems, including TSP, ATSP, CVRP, and PCTSP. Our code is available at: https://github.com/henry-yeh/GLOP.

Introduction

Routing problems pervade logistics, supply chain, transportation, robotic systems, etc. Modern industries have witnessed ever-increasing demands for the massive and expeditious routing of goods, services, and people. The traditional solvers based on mathematical programming or iterative heuristics struggle to keep pace with such growing complexity and real-time requirements.

Recent advances in Neural Combinatorial Optimization (NCO) (Bogyrbayeva et al. 2022) seek end-to-end solutions for routing problems, where neural solvers are exploited and empowered by massive training while enjoying potentially efficient inference. However, most existing NCO methods still struggle with real-time scaling-up performance; they are unable to solve routing problems involving thousands or tens of thousands of nodes in seconds, falling short of the need of modern industries (Hou et al. 2023).

In answer to that, this work proposes GLOP (**G**lobal and **L**ocal **O**ptimization **P**olicies) which partitions a large routing problem into sub-Travelling Salesman Problems (TSPs) and further partitions potentially large (sub-)TSPs into small Shortest Hamiltonian Path Problems (SHPPs). GLOP hybridizes non-autoregressive (NAR) global partition and autoregressive (AR) local construction policies, where the global policy learns the first partition and the local policy learns to solve SHPPs. We intend to integrate the strengths while circumventing the drawbacks of NAR and AR paradigms (further discussed in Appendix B). In particular, partitioning nodes into subsets (each corresponding to a TSP) well suits NAR heuristics, because it is a large-scale but coarse-grained task agnostic of within-subset node ordering. On the other hand, solving SHPPs could be efficiently handled with the AR heuristic because it is a small-scale but fine-grained task.

The solution pipeline of GLOP is applicable to variations of routing problems, such as those tackled in (Li et al. 2021a; Zhang et al. 2022a, 2021; Alesiani, Ermis, and Gkiotsalitis 2022; Miranda-Bront et al. 2017; Li et al. 2021b). We evaluate GLOP on canonical TSP, Asymmetric TSP (ATSP), Capacitated Vehicle Routing Problem (CVRP), and Prize Collecting TSP (PCTSP). GLOP for (A)TSP, as opposed to most methods that require scale-specific and distribution-specific training, can perform consistently and competitively across scales, across distributions, and on real-world benchmarks, using the same set of local policies. Notably, it is the first neural solver to effectively scale to TSP100K, obtaining a 5.1% optimality gap and a $174\times$ speedup compared with 1-run 1-trial LKH-3. GLOP for CVRP clearly outperforms prior state-of-the-art (SOTA) real-time solvers (Hou et al. 2023) while using $10\times$ less execution time. On PCTSP, GLOP surpasses both recent neural solvers and conventional solvers.

Accordingly, we summarize our contributions as follows:

- We propose GLOP, a versatile framework that extends existing neural solvers to large-scale problems. To our knowledge, it makes the first effective attempt at hybridizing NAR and AR end-to-end NCO paradigms.
- We propose to learn global partition heatmaps for decomposing large-scale routing problems, leveraging NAR heatmap learning in a novel way.
- We propose a one-size-fits-all real-time (A)TSP solver that learns small SHPP solution construction for arbitrarily large (A)TSP. We dispense with learning upper-level TSP policies suggested in (Kim, Park, and Kim 2021; Pan et al. 2023) while achieving better performance.

*Corresponding author.

- On (A)TSP, GLOP delivers competitive scaling-up and cross-distribution performance and is the first neural solver to scale to TSP100K effectively. On CVRP and PCTSP, GLOP achieves SOTA real-time performance.

Background and Related Work

Neural Combinatorial Optimization (NCO)

Recent advances in NCO show promise for solving combinatorial optimization problems in an end-to-end manner (Bogyrbayeva et al. 2022; Mazyavkina et al. 2021; Berto et al. 2023). The end-to-end neural routing solvers can be categorized into two paradigms: AR solution construction and NAR heatmap generation coupled with subsequent decoding. We defer further discussions to Appendix C.

Divide and Conquer for VRP

The idea of "divide and conquer" has long been applied to VRP variants in traditional (meta) heuristics (Zhang et al. 2021; Alesiani, Ermis, and Gkiotsalitis 2022; Xiao et al. 2019; Taillard and Helsgaun 2019). Recently, such an idea has been introduced in neural routing solvers. Li, Yan, and Wu (2021) propose learning to delegate (L2D) the improvement of subtours to LKH-3 (Helsgaun 2017). Zong et al. (2022) introduce Rewriting-by-Generating (RBG) framework that involves repeated learning-based merging and rule-based decomposition. However, both methods rely on iterative refinement, therefore holding back the real-time performance. More related to GLOP, Hou et al. (2023) present a Two-stage Divide Method (TAM), the prior SOTA real-time neural solver for large-scale CVRP, where a dividing model learns to partition CVRP into sub-TSPs autoregressively, and the sub-TSPs are then solved by sub-solvers such as Attention Model (AM) (Kool, van Hoof, and Welling 2019) or LKH-3. Unlike other prior works, both TAM and GLOP target large-scale CVRP under real-time settings. By comparison, GLOP outperforms TAM on CVRP by leveraging more effective global representations and better neural sub-TSP solvers, and can also handle routing problems that TAM does not address.

Local Construction for TSP

Learning local subtour reconstruction for TSP is initially introduced by Kim, Park, and Kim (2021) in Learning Collaborative Policy (LCP). LCP generates diversified initial solutions (seeds) with neural models (seeders), then repeatedly decomposes and reconstructs them. However, LCP, limited mainly by the design of seeders, can hardly scale up to TSP with hundreds of nodes. More recently, Pan et al. (2023) propose H-TSP, a hierarchical TSP solver interleaving forming open-loop TSP (a.k.a., SHPP) with upper-level policies and conquering it. By comparison, GLOP dispenses with learning any upper-level TSP policy but is able to outperform H-TSP. Another concurrent work, namely select-and-optimize (SO) (Cheng et al. 2023), utilizes a TSP solution pipeline similar to GLOP. But SO heavily relies on sophisticated heuristics specific to TSP, resulting in prolonged computational time. By comparison, GLOP achieves competitive solutions while being hundreds of times more efficient.

Methodology Overview

GLOP is schematically illustrated in Figure 1. It aims to provide a unified and scalable framework for heterogeneous vehicle routing problems. To this end, our design targets three representative problem settings: (1) large-scale TSP alone, (2) large-scale CVRP requiring problem partitioning and solving multiple small sub-TSPs, and (3) large-scale PCTSP requiring problem partitioning and solving a single large sub-TSP. We defer the detailed explanations of these problems to Appendix D.

In general, GLOP learns local policies for (sub-)TSP and global policies for partitioning general routing problems into sub-TSPs. Our (sub-)TSP solver generates initial TSP tours using Random Insertion, divides the complete tours into independent subtours, and learns to reconstruct them for improvements. Our general routing solver additionally learns to perform node clustering or subsetting that generates sub-TSP(s). We elaborate on our local policy and global policy below and provide more details in Appendix A.

(Sub-)TSP Solver

Inference Pipeline

GLOP learns local policies to improve a TSP solution by decomposing and reconstructing it.

Initialization GLOP generates an initial TSP tour with Random Insertion (RI), a simple and generic heuristic. RI greedily picks the insertion place that minimizes the insertion cost for each node.

Then, GLOP performs improvements on the initial tour. Following Kim, Park, and Kim (2021), we refer to a round of improvement as a "revision"; we refer to a local policy parameterized by an autoregressive NN and trained to solve SHPPn (SHPP of n nodes) as "Reviser-n". A revision involves decomposing and reconstructing the initial tour, comprising four sequential steps outlined below.

Decomposition When improved by Reviser-n, a complete tour with N nodes is randomly decomposed into $\lfloor \frac{N}{n} \rfloor$ subtours, each with n nodes. There is no overlap between every two subtours. A "tail subtour" with $N \bmod n$ nodes, if any, is left untouched until composition. Each subtour corresponds to an SHPP graph, and reconstructing a subtour is equivalent to solving an SHPP instance. We pick the decomposition positions uniformly when performing repeated revisions.

Transformation and augmentation To improve the predictability and homogeneity of the model inputs, we apply Min-max Normalization and an optional rotation to the SHPP graphs. They scale the x-axis coordinates to the range $[0, 1]$ and set the lower bound of the y-axis to 0. In addition, we augment the SHPP instances by flipping the node coordinates to enhance the model performance.

Solving SHPPs with local policies We autoregressively reconstruct the subtours (i.e., solve the SHPP instances) with trainable revisers. Any SHPP solutions that are worse than the current ones will be discarded. This key step is detailed below.

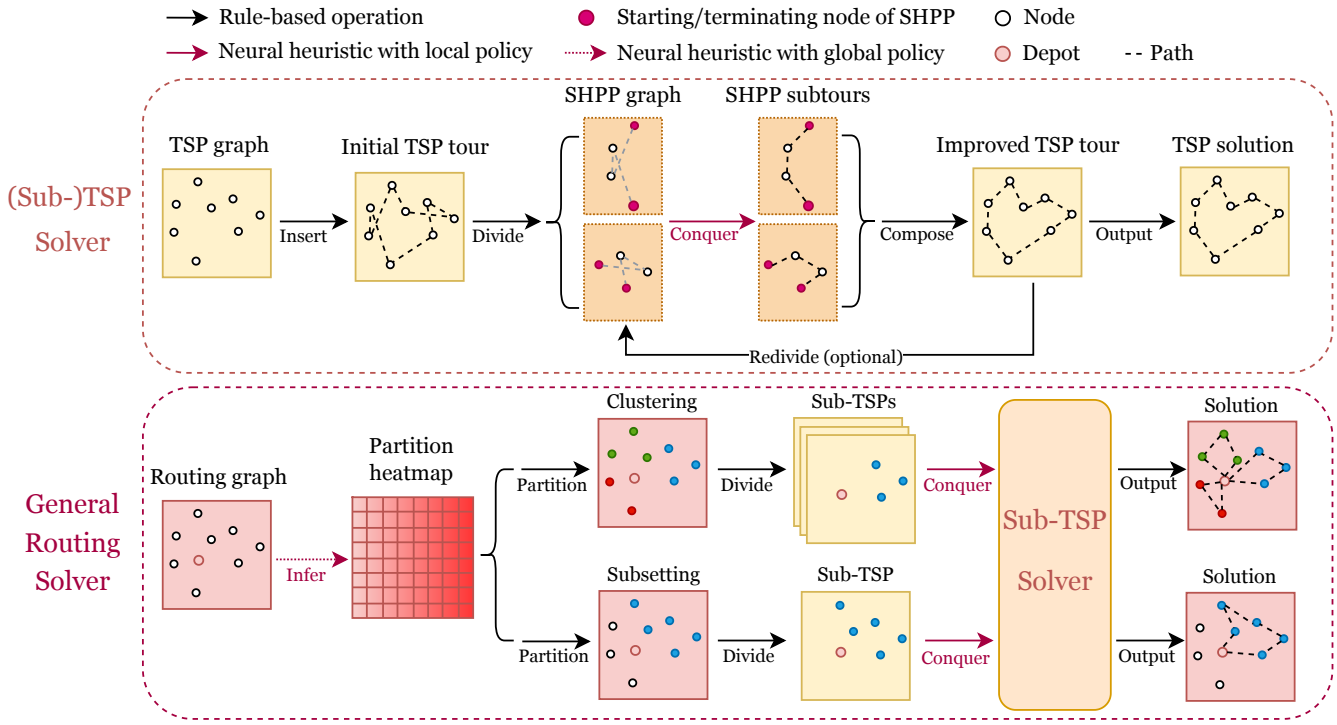

Figure 1: The pipeline of GLOP.

Composition The $\lfloor \frac{N}{n} \rfloor$ reconstructed (or original) subtours and a tail subtour, if any, compose an improved complete tour by connecting the starting/terminating nodes of SHPPs in their original order.

GLOP can apply multiple revisers to solve a problem from different angles. Also, a single reviser can decompose the tour at different points and repeat its revisions. After all revisions, GLOP outputs the improved tour as its final solution. Notably, GLOP allows applying a single set of small-SHPP-trained models for arbitrarily large TSP.

Solving SHPP with Local Policy

Problem formulation and motivation SHPP is also referred to as open-loop TSP. With starting/terminating nodes fixed, it aims to minimize the length of a Hamiltonian path visiting all nodes in between exactly once. Solving small SHPPs with neural networks, instead of directly solving TSP or with traditional heuristics, makes GLOP a highly parallelizable one-size-fits-all solution.

Model We parameterize our local policies based on Attention Model (AM) (Kool, van Hoof, and Welling 2019). To apply it to SHPP, we adjust its context embedding following Kim, Park, and Kim (2021) and leverage the solution symmetries by autoregressively constructing solutions from both starting/determining nodes.

Local policy Given an SHPP instance $\boldsymbol{s}$ with starting/terminating node 1 and n, our stochastic local policy $p_{\boldsymbol{\theta}}(\boldsymbol{\omega}_{fd}, \boldsymbol{\omega}_{bd} \mid \boldsymbol{s})$, parameterized by the neural model $\boldsymbol{\theta}$, denotes the conditional probability of constructing forward and backward-decoded solutions $\boldsymbol{\omega}_{fd}$ and $\boldsymbol{\omega}_{bd}$. We let $\boldsymbol{\omega}_{1:t-1}$ denote the partial solution at time step t, then the local policy can be factorized into probability distribution of per-step construction:

$$\begin{aligned} p_{\boldsymbol{\theta}}(\boldsymbol{\omega}_{fd}, \boldsymbol{\omega}_{bd} \mid \boldsymbol{s}) &= p_{\boldsymbol{\theta}}(\boldsymbol{\omega}_{fd} \mid \boldsymbol{s}) \times p_{\boldsymbol{\theta}}(\boldsymbol{\omega}_{bd} \mid \boldsymbol{s}) \\ &= \prod_{t=1}^{n-2} p_{\boldsymbol{\theta}}(\omega_t \mid \boldsymbol{s}, \boldsymbol{\omega}_{1:t-1}, n) \times p_{\boldsymbol{\theta}}(\omega_t \mid \boldsymbol{s}, \boldsymbol{\omega}_{1:t-1}, 1). \end{aligned} \tag{1}$$

We accept the better one between $\boldsymbol{\omega}_{fd}$ and $\boldsymbol{\omega}_{bd}$ during inference while making use of both for training.

Training Algorithm

We train our parameterized local policy, i.e., a reviser, by minimizing the expected length of its constructed SHPP solutions:

$$\begin{aligned} \text{minimize} \quad \mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{s}) = \mathbb{E}_{\boldsymbol{\omega}_{fd}, \boldsymbol{\omega}_{bd} \sim p_{\boldsymbol{\theta}}(\boldsymbol{\omega}_{fd}, \boldsymbol{\omega}_{bd} \mid \boldsymbol{s})} \\ [f_{SHPP}(\boldsymbol{\omega}_{fd}, \boldsymbol{s}) + f_{SHPP}(\boldsymbol{\omega}_{bd}, \boldsymbol{s})], \end{aligned} \tag{2}$$

where f_{SHPP} maps an SHPP solution to its length. We apply the REINFORCE-based gradient estimator (Williams 1992) using the average path length of two greedy rollouts as a baseline. This training algorithm doubles the experience learned on each instance and enables a more reliable baseline by weighing the greedy rollouts of both directions.

Two-stage curriculum learning According to our coordinate transformation, we design a two-stage curriculum to improve the homogeneity and consistency between training and inference instances. We are motivated by the following

observation: the inputs to revisers are 1) SHPP graphs with y-axis upper bounds ranging from 0 to 1 after our coordinate transformation, and also 2) the outputs of its preceding module. Therefore, stage 1 in our curriculum trains revisers using multi-distribution SHPPs with varied y-axis upper bounds, and stage 2 collaboratively fine-tunes all revisers following the inference pipeline.

General Routing Solver

Many routing problems can be formulated hierarchically, which requires node clustering (e.g. CVRP, mTSP, Capacitated Arc Routing Problem) or node subsetting (e.g. PCTSP, Orienteering Problem, Covering Salesman Problem), followed by solving multiple sub-TSPs or a single sub-TSP, respectively (Li et al. 2021a; Zhang et al. 2022a, 2021; Alesiani, Ermis, and Gkiotsalitis 2022; Xiao et al. 2019; Li et al. 2021b). For these general routing problem, GLOP involves an additional global partition policy defined by a parameterized partition heatmap and trained with parallel on-policy sampling without costly step-by-step neural decoding. The applicability of our global policy is also discussed.

Global Policy as Partition Heatmap

Partition heatmap We introduce a parameterized partition heatmap $\boldsymbol{\mathcal{H}}_{\boldsymbol{\phi}}(\boldsymbol{\rho}) = [h_{ij}(\boldsymbol{\rho})]_{(n+1)\times(n+1)}$ where $\boldsymbol{\rho}$ is the input instance with $n+1$ nodes including node 0 as the depot. $h_{ij} \in \mathbb{R}^+$ represents the unnormalized probability of nodes i and j belonging to the same subset.

Model and input graph The partition heatmap is parameterized by an isomorphic GNN $\boldsymbol{\phi}$ (Joshi, Laurent, and Bresson 2019; Qiu, Sun, and Yang 2022). Inputs to the model are sparsified graphs with features designed separately for different problems. We defer the full details to Appendix A.4.

Global policy For node clustering, GLOP partitions all nodes into multiple subsets, each corresponding to a sub-TSP to solve. For node subsetting, GLOP partitions all nodes into two subsets, i.e., the to-visit subset and the others, where the to-visit subset forms a sub-TSP to solve. Let $\boldsymbol{\pi} = \{\boldsymbol{\pi}^r\}_{r=1}^{|\boldsymbol{\pi}|}$ denote a complete partition and $\boldsymbol{\pi}^r = \{\pi_t^r\}_{t=1}^{|\boldsymbol{\pi}^r|}$ the r-th subset containing both regular nodes and the depot. Each subset begins and terminates at the depot; that is, $\pi_1^r = \pi_{|\boldsymbol{\pi}^r|}^r = 0$. Given $\boldsymbol{\mathcal{H}}_{\boldsymbol{\phi}}(\boldsymbol{\rho})$, our global policy partitions all nodes into $|\boldsymbol{\pi}|$ subsets by sequentially sampling nodes while satisfying problem-specific constraints Θ:

$$p_{\boldsymbol{\phi}}(\boldsymbol{\pi}|\boldsymbol{\rho}) = \begin{cases} \prod_{r=1}^{|\boldsymbol{\pi}|} \prod_{t=1}^{|\boldsymbol{\pi}^r|-1} \frac{h_{\pi_t^r,\pi_{t+1}^r}(\boldsymbol{\rho})}{\sum_{k\in\mathcal{N}(\boldsymbol{\pi}^p)} h_{\pi_t^r,k}(\boldsymbol{\rho})}, & \text{if } \boldsymbol{\pi} \in \Theta, \\ 0, & \text{otherwise,} \end{cases} \tag{3}$$

where $\mathcal{N}(\boldsymbol{\pi}^p)$ is the set of feasible actions given the current partial partition. For our benchmark problems, the applied constraints Θ are given in Appendix A.4.

Decoding We apply greedy (GLOP-G) and sampling (GLOP-S) heatmap decoding (Qiu, Sun, and Yang 2022) to draw the node partitions following our global policy.

Training Algorithm

We train our global policy to output partitions that could lead to the best-performing final solutions after solving sub-TSPs. Given each instance $\boldsymbol{\rho}$, the training algorithm infers partition heatmap $\boldsymbol{\mathcal{H}}_{\boldsymbol{\phi}}(\boldsymbol{\rho})$, samples node partitions in parallel, feeds the sampled partitions into GLOP for sub-TSP solutions, and optimizes the expected final performance:

$$\min \mathcal{L}(\boldsymbol{\phi}|\boldsymbol{\rho}) = \mathbb{E}_{\boldsymbol{\pi}\sim p_{\boldsymbol{\phi}}(\boldsymbol{\pi}|\boldsymbol{\rho})}[\sum_{r=1}^{|\boldsymbol{\pi}|} f_{TSP}(GLOP_{\boldsymbol{\theta}}(\boldsymbol{\pi}^r, \boldsymbol{\rho}))], \tag{4}$$

where f_{TSP} is a mapping from a sub-TSP solution to its length, and $GLOP_{\boldsymbol{\theta}}$ generates sub-TSP solutions with well-trained local policies. We apply the REINFORCE algorithm (Williams 1992) with the averaged reward of sampled solutions for the same instance as a baseline. The baseline is respectively computed for each instance within each training batch. GLOP for sub-TSP, i.e. $GLOP_{\boldsymbol{\theta}}$, enables efficient training of our global policy on large-scale problems due to its parallelizability and scalability.

Applicability

Many routing problems can be formulated hierarchically, involving node clustering and/or node subsetting, depending on the problem formulation. Node clustering is used when the problem requires formulating multiple routes that cover all nodes, while node subsetting is used when the problem requires formulating a single route that covers a subset of nodes. In some cases, a routing problem may require both subsetting and clustering. Our global policy offers a unified formulation for all these scenarios. Additionally, it can easily handle constraints via masking if they can be anticipated while constructing node subsets. To handle more complex constraints, one can assign a large negative value to the rewards of infeasible solutions or apply post-processing techniques before solution evaluation.

Experimentation

Experimental Setup

Datasets We refer the readers to Kool, van Hoof, and Welling (2019) for more specific definitions of benchmark problems. (1) We evaluate GLOP on uniformly sampled large-scale TSP instances (i.e., TSP500, 1K, and 10K) used in Fu, Qiu, and Zha (2021) and an additionally generated TSP100K instance. We perform a cross-distribution evaluation on test instances used in (Bi et al. 2022). For evaluation on real-world benchmarks, we draw all 49 symmetric TSP instances featuring *EUC_2D* and containing fewer than 1000 nodes (since most baselines cannot process larger-scale instances) from TSPLIB and map all instances to the $[0, 1]^2$ square through Min-max Normalization. The test datasets of ATSP are generated following Kwon et al. (2021). (2) For CVRP, we adhere to the settings in TAM (Hou et al. 2023) and use the code of AM (Kool, van Hoof, and Welling 2019) to generate test datasets on CVRP1K, 2K, 5K, and 7K, each containing 100 instances. We also evaluate GLOP on several large-scale CVRPLIB instances. (3) For PCTSP,

Method	TSP500			TSP1K			TSP10K		
	Obj.	Gap(%)	Time	Obj.	Gap(%)	Time	Obj.	Gap(%)	Time
Concorde	16.55	0.00	40.9m	23.12	0.00	8.2h	-	-	-
LKH-3	16.55	0.00	5.5m	23.12	0.00	24m	71.77	0.00	13h
Random Insertion	18.59	12.3	<1s	26.12	13.0	<1s	81.84	14.0	5.7s
AM	22.60	36.6	5.8m	42.53	84.0	22m	430	499	3.5m
LCP	20.82	25.8	29m	36.34	57.2	34m	357	397	4.3m
GCN+MCTS ×12	16.96	2.48	2.4m+33s	23.86	3.20	4.9m+1.2m	75.73	5.50	7.1m+6.0m
POMO-EAS	24.04	45.3	1.0h	47.79	107	8.6h		OOM	
DIMES+S	19.06	15.0	2.0m	26.96	16.1	2.4m	86.25	20.0	3.1m
DIMES+MCTS ×12	17.01	2.78	1.0m+2.1m	23.86	3.20	2.6m+1.0m	76.02	5.90	13.7m+20m
Tspformer*	17.57	5.97	3.1m	27.02	16.9	5.0m	-	-	-
H-TSP	-	-	-	24.65	6.62	47s	77.75	7.32	48s
Pointerformer	17.14	3.56	1.0m	24.80	7.30	6.5m	-	-	-
DeepACO	16.94	2.36	4.3m	23.85	3.16	1.1h	-	-	-
GLOP	17.07	3.14	**19s**	24.01	3.85	**34s**	75.62	5.36	**32s**
GLOP (more revisions)	**16.91**	**1.99**	1.5m	**23.84**	**3.11**	3.0m	**75.29**	**4.90**	1.8m

Table 1: Comparison results on 128 TSP500, 128 TSP1K, and 16 TSP10K. For all experiments on TSP, "Time" is the total runtime for solving all instances. If it has two terms, they correspond to the runtime of heatmap generation and MCTS, respectively. OOM: out of our graphics memory (24GB). *: Results are drawn from the original literature with runtime proportionally adjusted (128/100) to match the size of our test datasets. See Appendix A.6 and F for full implementation details of GLOP and the baselines, respectively.

Method	TSP100K		
	Obj.	Gap(%)	Time
LKH-3 $\{T = 1\}$	226.4	0.00	8.1h
Random Insertion	258.5	14.2	1.7m
AM		OOM	
LCP		OOM	
GCN+MCTS ×12		OOM	
POMO-EAS		OOM	
DIMES+MCTS ×12		OOM	
DIMES+S	286.1	26.4	2.0m
H-TSP		OOM	
Pointerformer		OOM	
GLOP	240.0	6.01	**1.8m**
GLOP (more revisions)	**238.0**	**5.10**	2.8m

Table 2: Comparison results on a TSP100K instance.

Method	Avg. gap(%)	Time
LCP $\{M = 1280\}$	99.9	3.6m
DACT $\{T = 1K\}$	865	50m
GCN+MCTS ×1	1.10	7.5m
POMO-EAS $\{T = 60\}$	18.8	20m
DIMES+MCTS ×1	2.21	7.4m
AMDKD+EAS $\{T = 100\}$	7.86*	48m
Pointerformer	6.04	48s
GLOP	1.53	**42s**
GLOP (more revisions)	**0.69**	2.6m

*: Two instances are skipped due to OOM issue.

Table 3: Comparison results on TSPLIB instances.

Method	MatNet (Kwon et al. 2021)	GLOP
ATSP150	2.88 (7.2s)	**1.89 (6.4s)**
ATSP250	4.49 (12s)	**2.10 (9.6s)**
ATSP1000	-	**2.79 (39s)**

Table 4: Comparison results on ATSP.

we follow the settings in AM (Kool, van Hoof, and Welling 2019) for data generation on PCTSP500, 1K, and 5K. As suggested by Kool, van Hoof, and Welling (2019), we specify $K^n = 9, 12, 20$ to sample prizes for n =500, 1K, 5K, respectively , i.e., $\beta_i \sim \text{Uniform}(0, 3\frac{K^n}{n})$.

Baselines Evaluating an NCO method typically involves two metrics: the objective value and the runtime. To ensure the validity of our comparisons, we select SOTA baselines with adjustable runtime that can match GLOP. We defer detailed implementations of the baselines to Appendix F.

Hardware Unless otherwise stated, GLOP and the baselines are executed on a 12-core Intel(R) Xeon(R) Platinum 8255C CPU and an NVIDIA RTX 3090 Graphics Card.

Travelling Salesman Problem

Large-scale TSP Comparison results on large-scale TSP are shown in Table 1 and Table 2. For GCN+MCTS (Fu, Qiu, and Zha 2021) and DIMES+MCTS (Qiu, Sun, and Yang 2022), we use all 12 CPU cores for MCTS and limit its running time to ensure comparable results. Concluded from the results, GLOP is highly efficient due to its decomposed solution scheme (see Appendix E.5 for the analysis of time complexity). Furthermore, the memory consumption of

Method	Time	Uniform	Expansion		Explosion		Implosion	
		Gap(%)	Gap(%)	Det.(%)	Gap(%)	Det.(%)	Gap(%)	Det.(%)
AM (Kool, van Hoof, and Welling 2019)	0.5h	2.310	17.97	678	3.817	65	2.431	5.2
AM+HAC (Zhang et al. 2022b)	0.5h	2.484	3.997	**61**	3.084	24	2.595	4.5
GLOP		**0.091**	**0.166**	82	**0.066**	**-27**	**0.082**	**-9.9**
AMDKD+EAS (Bi et al. 2022)	2.0h	0.078	0.165	112	0.048	-39	0.079	1.3
GLOP (more revisions)		**0.048**	**0.076**	**60**	**0.028**	**-41**	**0.044**	**-8.3**

Table 5: Comparison results on the OoD datasets. Det $= \text{Gap}_{OoD}/\text{Gap}_U - 1$, where Gap_{OoD} and Gap_U are the optimality gaps on an OoD dataset and the Uniform dataset, respectively.

Method	CVRP1K		CVRP2K		CVRP5K		CVRP7K	
	Obj.	Time (s)	Obj.	Time (s)	Obj.	Time (s)	Obj.	Time (s)
LKH-3 (Helsgaun 2017)	46.4	6.2	64.9	20	175.7	152	245.0	501
AM (Kool, van Hoof, and Welling 2019)	61.4	0.6	114.4	1.9	257.1	12	354.3	26
L2I (Lu, Zhang, and Yang 2020)	93.2	6.3	138.8	25	-	-	-	-
NLNS (Hottung and Tierney 2019)	53.5	198	-	-	-	-	-	-
L2D (Li, Yan, and Wu 2021)	46.3	1.5	65.2	38	-	-	-	-
RBG (Zong et al. 2022)	74.0	13	137.6	42	-	-	-	-
TAM-AM (Hou et al. 2023)	50.1	0.8	74.3	2.2	172.2	12	233.4	26
TAM-LKH3 (Hou et al. 2023)	46.3	1.8	64.8	5.6	144.6	17	196.9	33
TAM-HGS (Hou et al. 2023)	-	-	-	-	142.8	30	193.6	52
GLOP-G	47.1	**0.4**	63.5	**1.2**	141.9	**1.7**	191.7	**2.4**
GLOP-G (LKH-3)	**45.9**	1.1	**63.0**	1.5	**140.6**	4.0	**191.2**	5.8

Table 6: Comparison results on large-scale CVRP following the settings in (Hou et al. 2023). "Time" corresponds to the per-instance runtime. GLOP-G (LKH-3) applies LKH-3 as its sub-TSP solver.

GLOP can be basically invariant of the problem scale if reconstructing the subtours using a fixed batch size. Hence, it is the first neural solver to effectively scale to TSP100K, obtaining a 5.1% optimality gap and a $174\times$ speed-up compared to LKH-3. Compared with LCP (Kim, Park, and Kim 2021) and H-TSP (Pan et al. 2023), GLOP dispenses with learning upper-level TSP policies while achieving better performance. Compared with NAR methods conducting MCTS refinements (Fu, Qiu, and Zha 2021; Qiu, Sun, and Yang 2022), GLOP generates reasonable solutions even before they have finished initialization or produced prerequisite heatmaps for solution decoding. Hence, GLOP exhibits clear advantages for real-time applications.

Cross-distribution TSP Table 5 gathers the comparison between GLOP and two baselines specially devised for cross-distribution performance (Zhang et al. 2022b; Bi et al. 2022) on four TSP100 datasets with different distributions, i.e., uniform, expansion, explosion, and implosion. Recall that we use uniformly distributed samples to train GLOP. Hence, the latter three datasets contain out-of-distribution (OoD) instances. Results show that GLOP obtains smaller gaps and less Det. on most OoD datasets. We argue that the holistic solution scheme of GLOP is the main contributor to its cross-distribution performance. The inputs to the neural models are local SHPP graphs, making GLOP insensitive to the overall TSP distribution. We conduct further discussions in Appendix E.3.

Real-world TSPLIB We evaluate GLOP and the baselines on real-world TSPLIB instances and collect the results in Table 3, where GLOP performs favorably against the baselines due to its consistent performance across scales and distributions. Note that we test each instance individually without parallel computation.

Asymmetric TSP GLOP is compatible with any neural architecture and can be extended to asymmetric distance. Table 4 exemplifies this flexibility on ATSP, where we replace AM with MatNet (Kwon et al. 2021) which is specially designed for ATSP. We follow the experimental setup in (Kwon et al. 2021) and generalize MatNet100 as baseline. For GLOP, we apply MatNet50 checkpoint as our reviser without retraining. The results validate that GLOP can successfully extend MatNet to solve large ATSP instances. Note that MatNet is limited to problem scales no larger than 256 due to its one-hot initialization while there is no such limitation for GLOP-empowered MatNet.

Capacitated Vehicle Routing Problem

GLOP for CVRP involves node clustering with our global policy, followed by solving the produced sub-TSPs with our sub-TSP solver.

Instance	Scale	AM		TAM-AM		LKH-3		TAM-LKH3		GLOP		GLOP-LKH3	
		Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time
LEUVEN1	3001	46.9	10s	20.2	10s	18.1	69s	19.3	16s	**16.9**	**2s**	**16.6**	**8s**
LEUVEN2	4001	53.3	13s	38.6	14s	22.1	74s	15.9	24s	21.8	**3s**	21.1	**3s**
ANTWERP1	6001	39.3	13s	24.9	13s	24.2	596s	24.0	25s	**20.3**	**3s**	**19.3**	**14s**
ANTWERP2	7001	50.3	15s	33.2	15s	31.1	479s	22.6	32s	**19.4**	**4s**	**19.4**	**7s**
GHENT1	10001	46.9	21s	30.2	22s	-		29.5	37s	**20.3**	**5s**	**18.3**	**22s**
GHENT2	11001	52.2	39s	33.3	38s	-		23.7	56s	**19.8**	**6s**	**18.1**	**8s**
BRUSSELS1	15001	52.4	131s	43.4	139s	-		27.2	167s	27.6	**8s**	27.5	**26s**
BRUSSELS2	16001	52.4	166s	39.0	159s	-		37.1	187s	**22.4**	**9s**	**20.1**	**14s**

Table 7: Comparison results on large-scale CVRPLIB instances.

Method	PCTSP500		PCTSP1K		PCTSP5K	
	Obj.	Time	Obj.	Time	Obj.	Time
OR Tools	15.0	1h	24.9	1h	63.3	1h
OR Tools (more iterations)	14.4	16h	20.6	16h	54.4	16h
AM (Kool, van Hoof, and Welling 2019)	19.3	14m	34.8	23m	175	21m
MDAM (Xin et al. 2021)	14.8	2.8m	22.2	17m	58.9	3h
GLOP-G	14.6	**26s**	20.0	**47s**	46.0	**3.7m**
GLOP-S	**14.3**	1.5m	**19.8**	2.5m	**44.9**	16m

Table 8: Comparison results of GLOP and the baselines on 128 PCTSP500, 1K, and 5K. "Time" corresponds to the total execution time for solving all instances.

Large-scale CVRP Table 6 summarizes the results of the comparison in large-scale CVRP, where the RBG (Zong et al. 2022) and L2D (Li, Yan, and Wu 2021) models are generalized for evaluation here due to their different training settings, and the results of other baselines are drawn from (Hou et al. 2023). Here, we apply the global policy trained on CVRP2K to both CVRP5K and CVRP7K, verifying the generalization performance of GLOP. Compared to the methods that entail iterative solution refinement (Helsgaun 2017; Li, Yan, and Wu 2021; Lu, Zhang, and Yang 2020; Zong et al. 2022), both TAM (Hou et al. 2023) and GLOP can deliver more real-time solutions. Compared with prior SOTA real-time solver TAM, GLOP learns more effective global/local policies and enables higher decoding efficiency. Hence, GLOP outperforms TAM regarding both solution quality and efficiency.

Real-world CVRPLIB We test GLOP on large-scale CVRPLIB instances and present results in Table 7. We generalize the model trained on CVRP2000 to these large instances and compare GLOP to prior SOTA real-time solver TAM (Hou et al. 2023). The results of TAM and other baselines are drawn from (Hou et al. 2023). The comparison also demonstrates the superiority of GLOP in both solution quality and efficiency, especially for very large instances.

Prize Collecting Travelling Salesman Problem

GLOP for PCTSP involves node subsetting with our global policies, followed by solving a sub-TSP instance with our sub-TSP solvers. Large-scale PCTSP entails both a scalable global policy to generate a promising node partition and a scalable local policy to tackle the equivalently large sub-TSP. We evaluate greedy (GLOP-G) and sampling (GLOP-S) decoding for our global policy. The comparison results on large-scale PCTSP are displayed in Table 8, where GLOP surpasses recent neural solvers and conventional solvers in terms of both solution quality and efficiency.

Conclusion and Limitation

This paper proposes GLOP to learn global policies for coarse-grained problem partitioning and local policies for fine-grained route construction. GLOP leverages the scalability of the NAR paradigm and meticulousness of the AR paradigm, making the first effective attempt at hybridizing them. Extensive evaluations on large-scale TSP, ATSP, CVRP, and PCTSP demonstrate its competitive and SOTA real-time performance. However, GLOP might be less competitive in application scenarios where prolonged execution time is allowed. In terms of its ability to trade off execution time for solution quality, GLOP might be inferior to the methods based on iterative solution refinement (further discussed in Appendix E.1). Our future focus will be on addressing this limitation. In addition, we plan to investigate the emerging possibilities that arise when viewing AR and NAR methods from a unified perspective. We believe it is also promising to exploit unsupervised Deep Graph Clustering techniques (Liu et al. 2022, 2023) or to formulate node classification tasks to solve large-scale routing problems hierarchically.

Acknowledgments

The authors appreciate the helpful discussions with Juntao Li, Yu Hong, and anonymous reviewers. Yu Hu, Jiusi Yin, and Tao Yu also contributed to this work. This research was supported by the National Natural Science Foundation of China (NSFC) [grant number 61902269, 62176172]; the National Key R&D Program of China [grant number 2018YFA0701700, 2018YFA0701701]; the Singapore Ministry of Education (MOE) Academic Research Fund (AcRF) Tier 1 grant; the Undergraduate Training Program for Innovation and Entrepreneurship, Soochow University [grant number 202210285001Z, 202310285041Z]; the Priority Academic Program Development of Jiangsu Higher Education Institutions, China; and Provincial Key Laboratory for Computer Information Processing Technology, Soochow University [grant number KJS1938].

References

Alesiani, F.; Ermis, G.; and Gkiotsalitis, K. 2022. Constrained Clustering for the Capacitated Vehicle Routing Problem (CC-CVRP). *Applied artificial intelligence*, 36(1): 1995658.

Berto, F.; Hua, C.; Park, J.; Kim, M.; Kim, H.; Son, J.; Kim, H.; Kim, J.; and Park, J. 2023. Rl4co: an extensive reinforcement learning for combinatorial optimization benchmark. *arXiv preprint arXiv:2306.17100*.

Bi, J.; Ma, Y.; Wang, J.; Cao, Z.; Chen, J.; Sun, Y.; and Chee, Y. M. 2022. Learning Generalizable Models for Vehicle Routing Problems via Knowledge Distillation. *ArXiv preprint*, abs/2210.07686.

Bogyrbayeva, A.; Meraliyev, M.; Mustakhov, T.; and Dauletbayev, B. 2022. Learning to Solve Vehicle Routing Problems: A Survey. *ArXiv preprint*, abs/2205.02453.

Cheng, H.; Zheng, H.; Cong, Y.; Jiang, W.; and Pu, S. 2023. Select and Optimize: Learning to solve large-scale TSP instances. In Ruiz, F.; Dy, J.; and van de Meent, J.-W., eds., *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, 1219–1231. PMLR.

Fu, Z.; Qiu, K.; and Zha, H. 2021. Generalize a Small Pretrained Model to Arbitrarily Large TSP Instances. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, 7474–7482. AAAI Press.

Helsgaun, K. 2017. An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems. *Roskilde: Roskilde University*, 24–50.

Hottung, A.; and Tierney, K. 2019. Neural large neighborhood search for the capacitated vehicle routing problem. *ArXiv preprint*, abs/1911.09539.

Hou, Q.; Yang, J.; Su, Y.; Wang, X.; and Deng, Y. 2023. Generalize Learned Heuristics to Solve Large-scale Vehicle Routing Problems in Real-time. In *The Eleventh International Conference on Learning Representations*.

Joshi, C. K.; Laurent, T.; and Bresson, X. 2019. An efficient graph convolutional network technique for the travelling salesman problem. *ArXiv preprint*, abs/1906.01227.

Kim, M.; Park, J.; and Kim, J. 2021. Learning Collaborative Policies to Solve NP-hard Routing Problems. In Ranzato, M.; Beygelzimer, A.; Dauphin, Y. N.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, 10418–10430.

Kool, W.; van Hoof, H.; and Welling, M. 2019. Attention, Learn to Solve Routing Problems! In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Kwon, Y.; Choo, J.; Yoon, I.; Park, M.; Park, D.; and Gwon, Y. 2021. Matrix encoding networks for neural combinatorial optimization. In Ranzato, M.; Beygelzimer, A.; Dauphin, Y. N.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, 5138–5149.

Li, J.; Ma, Y.; Gao, R.; Cao, Z.; Lim, A.; Song, W.; and Zhang, J. 2021a. Deep reinforcement learning for solving the heterogeneous capacitated vehicle routing problem. *IEEE Transactions on Cybernetics*, 52(12): 13572–13585.

Li, K.; Zhang, T.; Wang, R.; Wang, Y.; Han, Y.; and Wang, L. 2021b. Deep reinforcement learning for combinatorial optimization: Covering salesman problems. *IEEE transactions on cybernetics*, 52(12): 13142–13155.

Li, S.; Yan, Z.; and Wu, C. 2021. Learning to delegate for large-scale vehicle routing. In Ranzato, M.; Beygelzimer, A.; Dauphin, Y. N.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, 26198–26211.

Liu, Y.; Liang, K.; Xia, J.; Yang, X.; Zhou, S.; Liu, M.; Liu, X.; and Li, S. Z. 2023. Reinforcement Graph Clustering with Unknown Cluster Number. In *Proceedings of the 31st ACM International Conference on Multimedia*, 3528–3537.

Liu, Y.; Xia, J.; Zhou, S.; Wang, S.; Guo, X.; Yang, X.; Liang, K.; Tu, W.; Li, Z. S.; and Liu, X. 2022. A Survey of Deep Graph Clustering: Taxonomy, Challenge, and Application. *arXiv preprint arXiv:2211.12875*.

Lu, H.; Zhang, X.; and Yang, S. 2020. A Learning-based Iterative Method for Solving Vehicle Routing Problems. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Mazyavkina, N.; Sviridov, S.; Ivanov, S.; and Burnaev, E. 2021. Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research*, 134: 105400.

Miranda-Bront, J. J.; Curcio, B.; Méndez-Díaz, I.; Montero, A.; Pousa, F.; and Zabala, P. 2017. A cluster-first route-second approach for the swap body vehicle routing problem. *Annals of Operations Research*, 253: 935–956.

Pan, X.; Jin, Y.; Ding, Y.; Feng, M.; Zhao, L.; Song, L.; and Bian, J. 2023. H-TSP: Hierarchically Solving the Large-Scale Travelling Salesman Problem. *ArXiv preprint*, abs/2304.09395.

Qiu, R.; Sun, Z.; and Yang, Y. 2022. DIMES: A Differentiable Meta Solver for Combinatorial Optimization Problems. *ArXiv preprint*, abs/2210.04123.

Taillard, É. D.; and Helsgaun, K. 2019. POPMUSIC for the travelling salesman problem. *European Journal of Operational Research*, 272(2): 420–429.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Reinforcement learning*, 5–32.

Xiao, J.; Zhang, T.; Du, J.; and Zhang, X. 2019. An evolutionary multiobjective route grouping-based heuristic algorithm for large-scale capacitated vehicle routing problems. *IEEE transactions on cybernetics*, 51(8): 4173–4186.

Xin, L.; Song, W.; Cao, Z.; and Zhang, J. 2021. Multi-Decoder Attention Model with Embedding Glimpse for Solving Vehicle Routing Problems. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, 12042–12049. AAAI Press.

Zhang, R.; Zhang, C.; Cao, Z.; Song, W.; Tan, P. S.; Zhang, J.; Wen, B.; and Dauwels, J. 2022a. Learning to solve multiple-TSP with time window and rejections via deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*.

Zhang, Y.; Mei, Y.; Huang, S.; Zheng, X.; and Zhang, C. 2021. A route clustering and search heuristic for large-scale multidepot-capacitated arc routing problem. *IEEE Transactions on Cybernetics*, 52(8): 8286–8299.

Zhang, Z.; Zhang, Z.; Wang, X.; and Zhu, W. 2022b. Learning to Solve Travelling Salesman Problem with Hardness-Adaptive Curriculum. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, 9136–9144. AAAI Press.

Zong, Z.; Wang, H.; Wang, J.; Zheng, M.; and Li, Y. 2022. RBG: Hierarchically Solving Large-Scale Routing Problems in Logistic Systems via Reinforcement Learning. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 4648–4658.