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

Haoran Ye, Jiarui Wang, Helan Liang, Zhiguang Cao, Yong Li, Fanzhang Li

2024, AAAI

Contents

1. Background

2. Problem

3. Contribution Point

3. Methodology overview

4. Sub-TSP Solver

5. General Routing Solver

6. Experiment

7. Conclusion and Limitation

1. Background

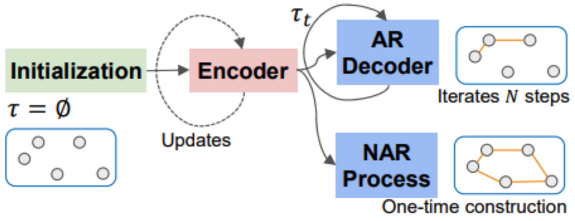


Fig. 3. Illustration of the generic construction process of L2C solvers, starting from an empty solution set and ending with complete solutions. Most L2C solvers are composed of an encoder and a decoder. Encoder is used to output the embeddings of VRP instances, while decoder selects nodes based on these embeddings to construct complete solutions.

L2C solver

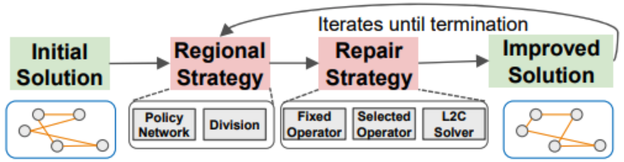


Fig. 4. Illustration of the iterative improvement solutions process of L2I solvers, starting from an initial complete solution and ending within a given timeframe. L2I solvers first rely on regional strategies to select regions (typically node pairs). Subsequently, diverse strategies are employed to repair the sub-tours of the selected regions.

L2I solver

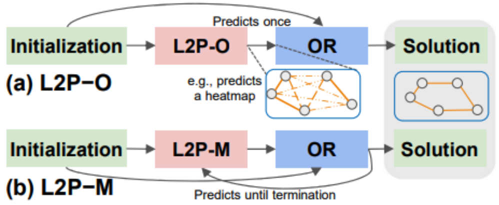


Fig. 6. Illustration of the information prediction processes of L2P-O and L2P-M solvers. L2P-O solvers aim to enhance OR algorithms by providing valuable information solely before the searches begin. Subsequently, OR algorithms address VRP instances by leveraging the predicted information along with problem definitions. Conversely, L2P-M solvers collaborate with OR algorithms continuously during the search processes, offering prediction of key information at each decision step. These L2P-M solvers take states of the OR algorithms, encompassing problem definitions, as their inputs [47].

L2P-O, L2P-M solver

reference: Neural Combinatorial Optimization Algorithms for Solving Vehicle Routing Problems: A Comprehensive Survey with Perspectives

1. Background

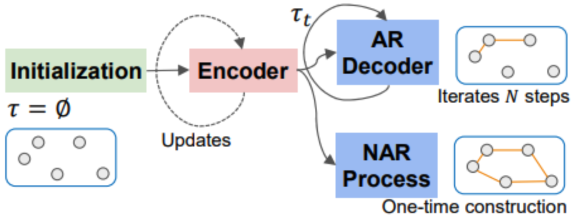


Fig. 3. Illustration of the generic construction process of L2C solvers, starting from an empty solution set and ending with complete solutions. Most L2C solvers are composed of an encoder and a decoder. Encoder is used to output the embeddings of VRP instances, while decoder selects nodes based on these embeddings to construct complete solutions.

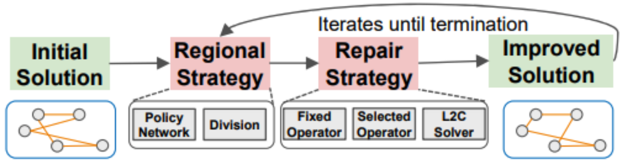


Fig. 4. Illustration of the iterative improvement solutions process of L2I solvers, starting from an initial complete solution and ending within a given timeframe. L2I solvers first rely on regional strategies to select regions (typically node pairs). Subsequently, diverse strategies are employed to repair the sub-tours of the selected regions.

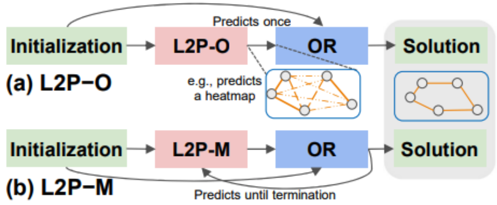


Fig. 6. Illustration of the information prediction processes of L2P-O and L2P-M solvers. L2P-O solvers aim to enhance OR algorithms by providing valuable information solely before the searches begin. Subsequently, OR algorithms address VRP instances by leveraging the predicted information along with problem definitions. Conversely, L2P-M solvers collaborate with OR algorithms continuously during the search processes, offering prediction of key information at each decision step. These L2P-M solvers take states of the OR algorithms, encompassing problem definitions, as their inputs [47].

L2C solver

L2I solver

L2P-O, L2P-M solver

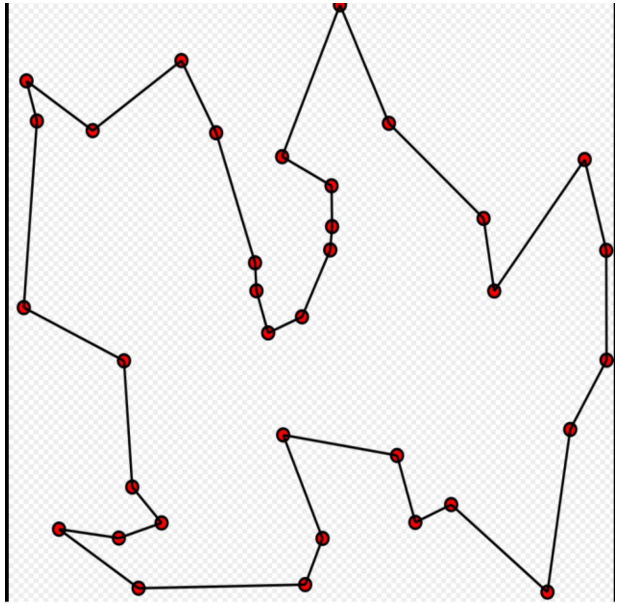
Well construct on small scale(100~1000)

Execution time , Solution quality tradeoff

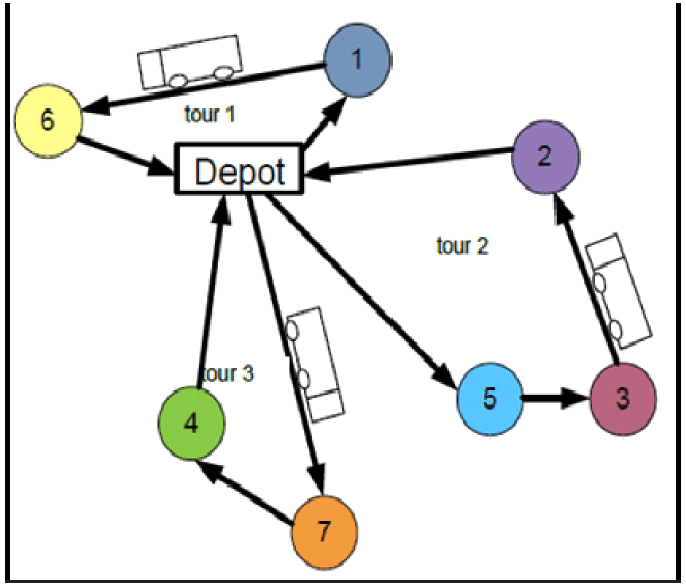
Execution time , Solution quality tradeoff

reference: Neural Combinatorial Optimization Algorithms for Solving Vehicle Routing Problems: A Comprehensive Survey with Perspectives

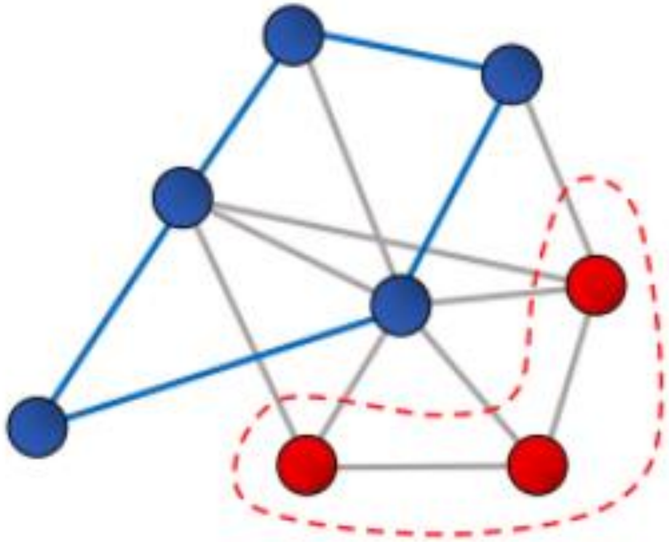
2. Problem



TSP, ATSP



CVRP



PCTSP

3. Contribution Point

- **GLOP, versatile framework** that extends existing **neural solvers to large-scale problems.**
  hybridizing NAR(Non-AutoRegressive) and AR(AutoRegressive) end to end NCO

- **To learn global partition heatmaps** for decomposing large-scale routing problems, leveraging **NAR heatmap** learning in a novel way

- propose a one-size-fits-all real-time (A)TSP solver that learns small SHPP solution construction for arbitrarily large (A)TSP

- 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

2. Methodology overview

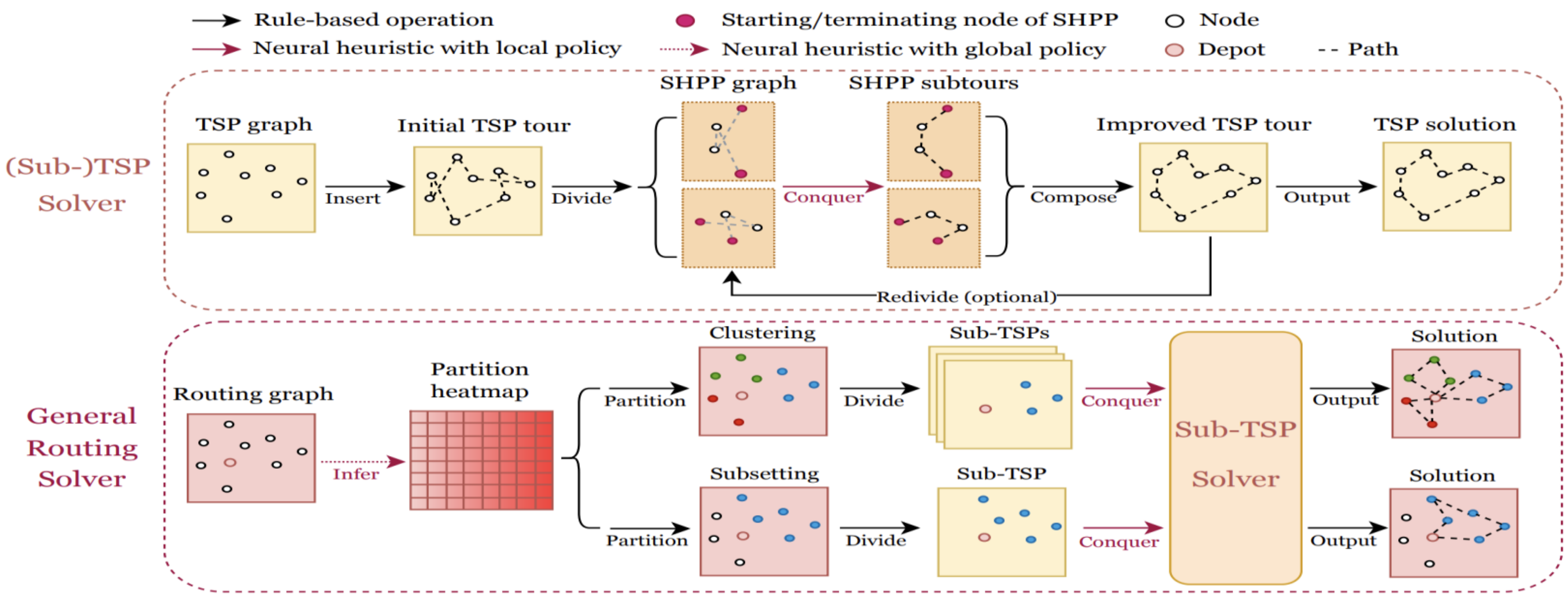
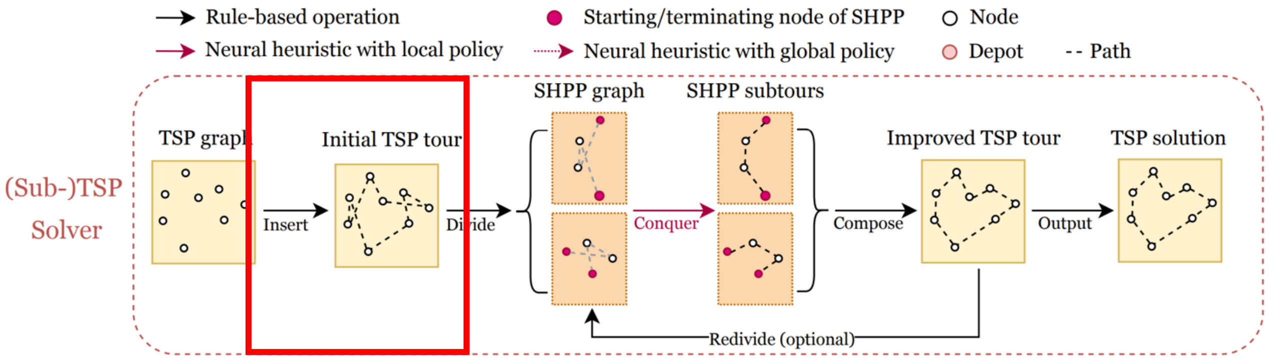


Figure 1: The pipeline of GLOP.

1. GLOP learns **local policies for (sub-)TSP**

2. GLOP learns **global policies for partitioning** general routing problems into sub-TSP

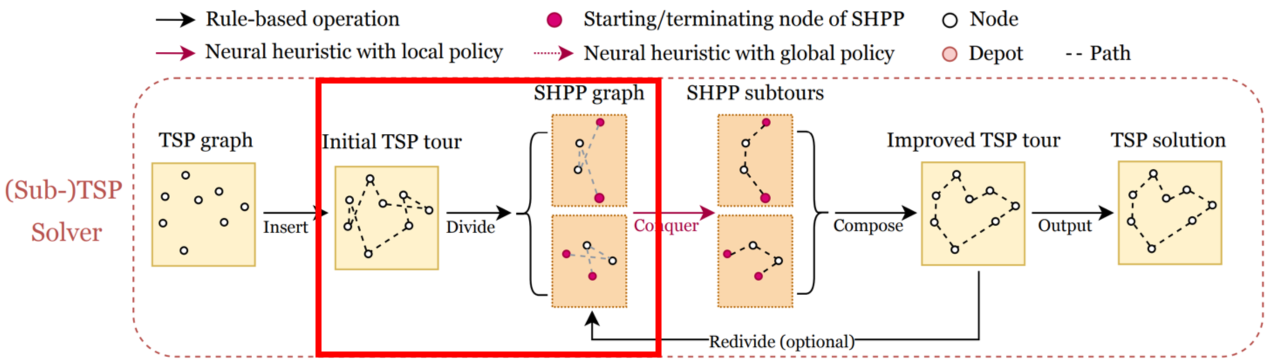
3. Sub-TSP solver – **Initialization**



Generate an initial TSP tour with Random Insertion(greedily)

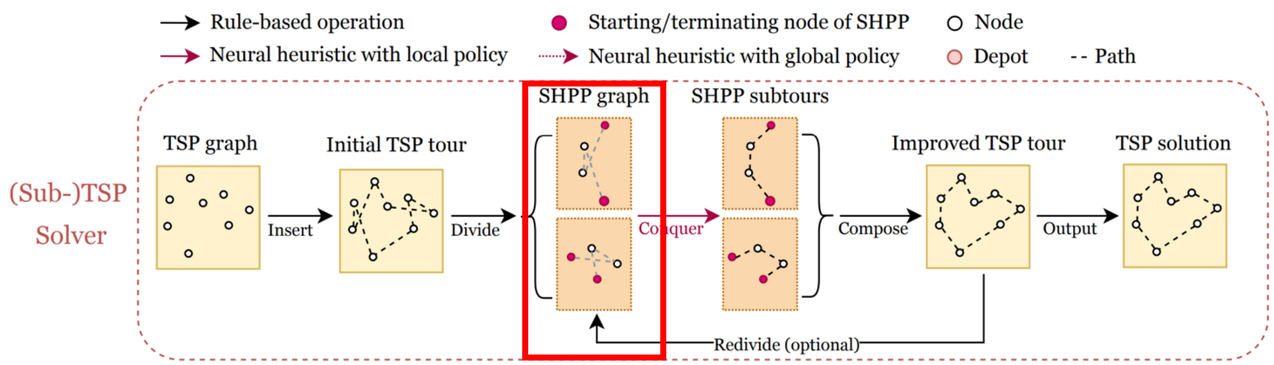
- Random insertion: The next node to join the tour, T, is selected randomly among the nodes still in (N - T) where N is the set of nodes of the network

3. Sub-TSP solver - **Decomposition**



Complete tour with N nodes is **randomly(uniformly)** decomposed into $\left[\frac{N}{M}\right]$subtours, each with n nodes

3. Sub-TSP solver - **Transformation and augmentation**



To improve predictability, homogeneity of the model inputs, apply Min-max Normalization, rotation(optional) to the SHPP graphs

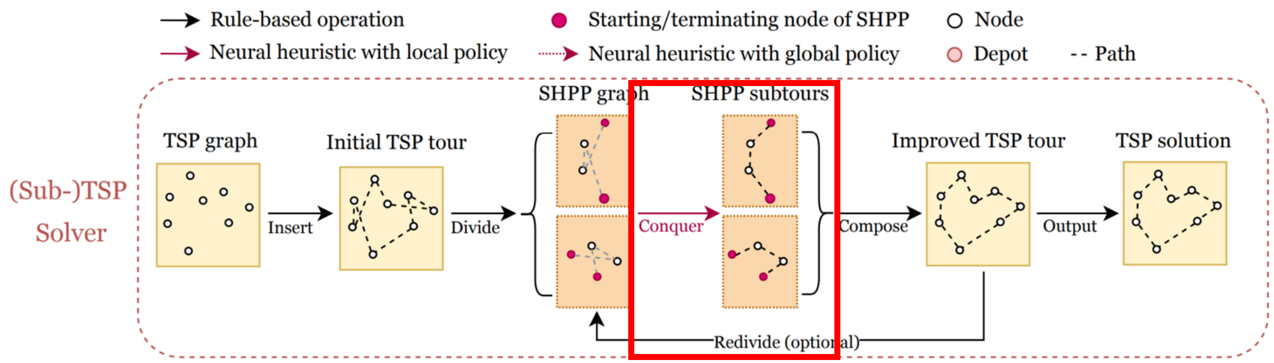
Min-max Normalization

$$x_i' = \begin{cases} sc(x_i - x_{\min}) & \text{if } x_{\max} - x_{\min} > y_{\max} - y_{\min}, \\ sc(y_i - y_{\min}) & \text{otherwise,} \end{cases}$$

$$y_i' = \begin{cases} sc(y_i - y_{\min}) & \text{if } x_{\max} - x_{\min} > y_{\max} - y_{\min}, \\ sc(x_i - x_{\min}) & \text{otherwise,} \end{cases}$$

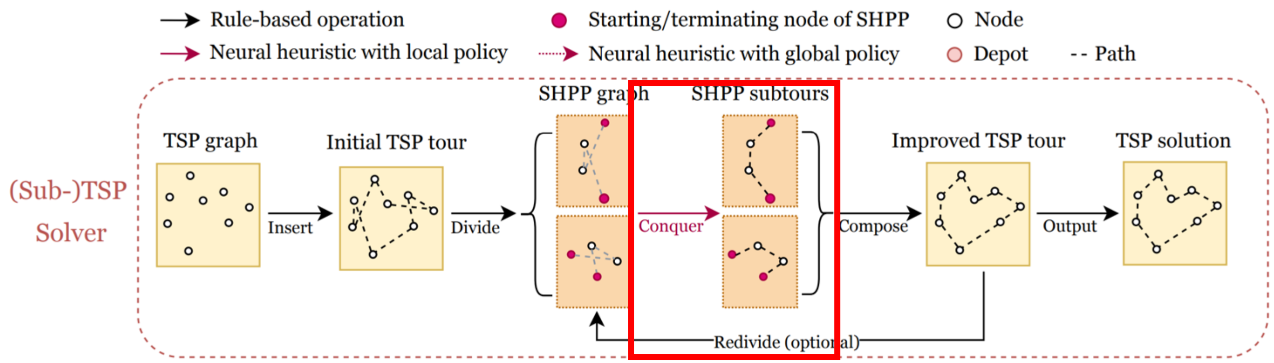
- $sc = \frac{1}{\max(x_{max} - x_{min}, y_{max} - y_{min})}$

3. Sub-TSP solver - **Solving SHPPs with local policies(Attention Model)**



Autoregressively reconstruct the subtours (i.e., solve the SHPP instances) with trainable revisers(LCP).

3. Sub-TSP solver - **Solving SHPPs with local policies(Attention Model)**



Training strategy:

consider the solution symmetries by autoregressive constructing solutions for both starting/determining nodes

❖ Local policy:

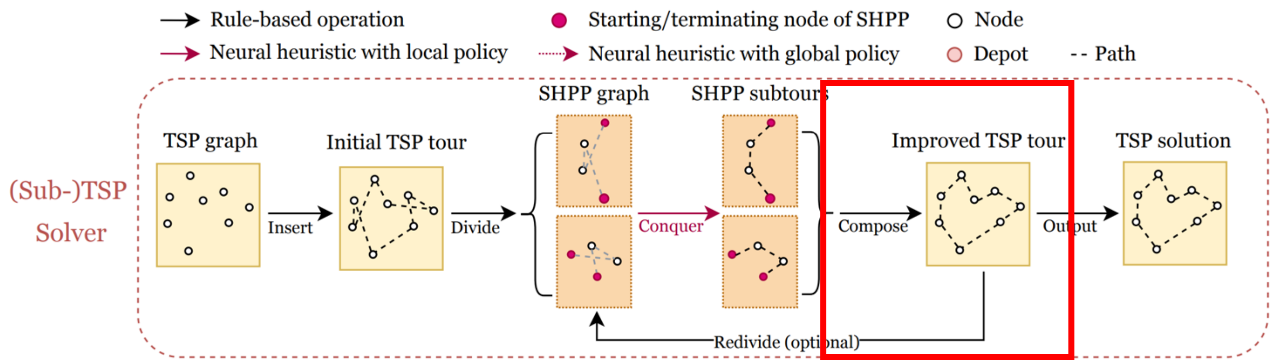
$$p_\theta(\omega_{fd}, \omega_{bd} \mid s) = p_\theta(\omega_{fd} \mid s) \times p_\theta(\omega_{bd} \mid s) = \prod_{t=1}^{n-2} p_\theta(\omega_t \mid s, \omega_{1:t-1}, n) \times p_\theta(\omega_t \mid s, \omega_{1:t-1}, 1)$$

- ω_{fd}: *forward construct solution*
- ω_{bd}: *backward construct solution*

❖ Training algorithm:

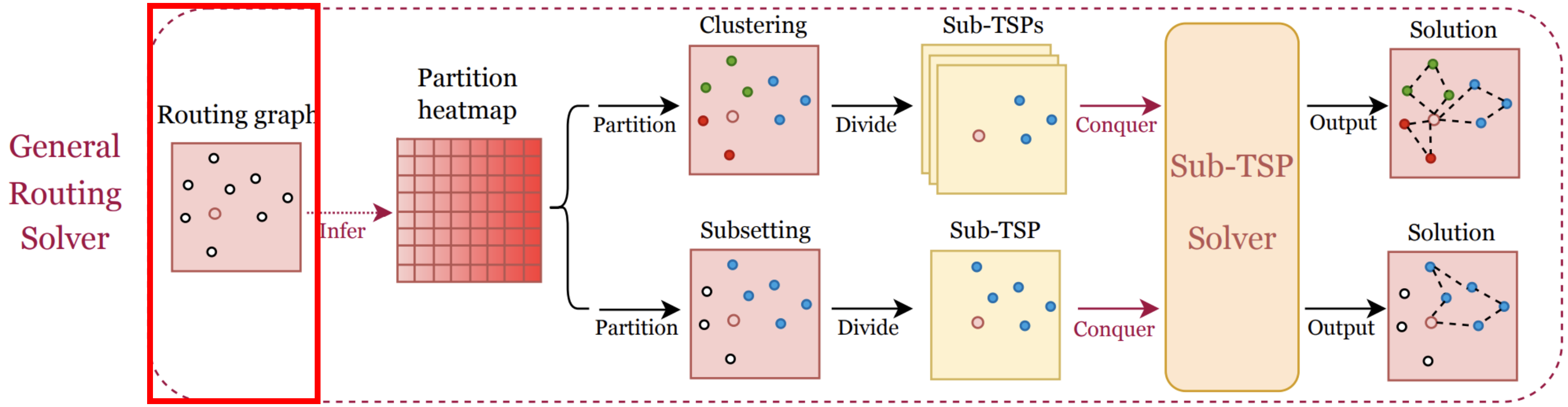
$$minimize\ \mathcal{L}(\theta|s) = \mathbb{E}_{\omega_{fd}, \omega_{bd} \sim p_\theta(\omega_{fd}, \omega_{bd}|s)}\left[f_{SHPP}(\omega_{fd}, s) + f_{SHPP}(\omega_{bd}, s)\right]$$

3. Sub-TSP solver - **Composition**



Compose an improved complete tour by connecting the starting/terminating nodes in their original order

4. General Routing Solver - **Model and Input graph**



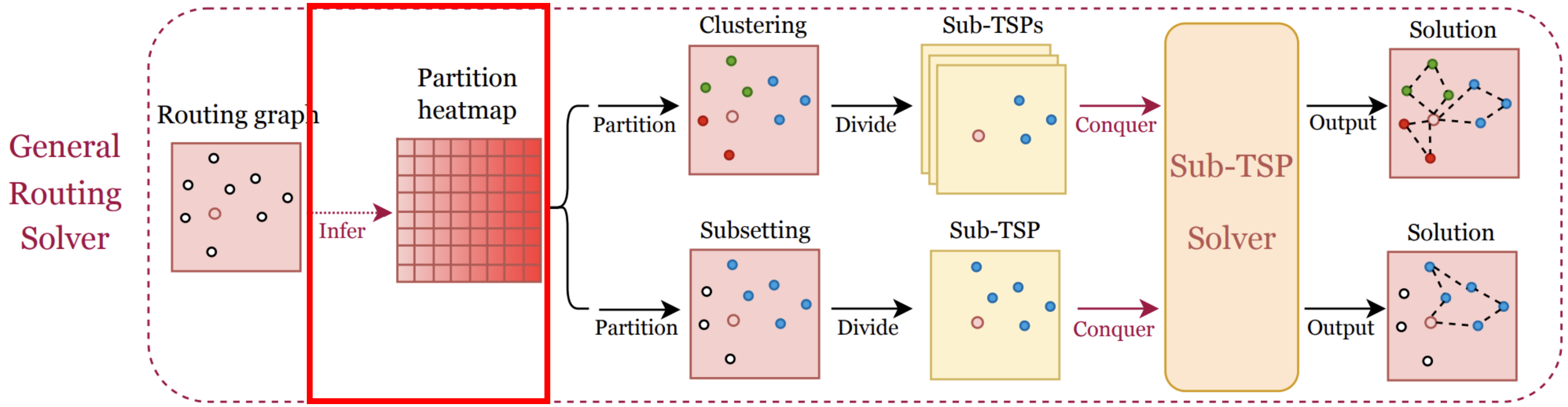
Input:

sparsified graphs with feature

Model:

isomorphic GNN

4. General Routing Solver - **Partition Heatmap**



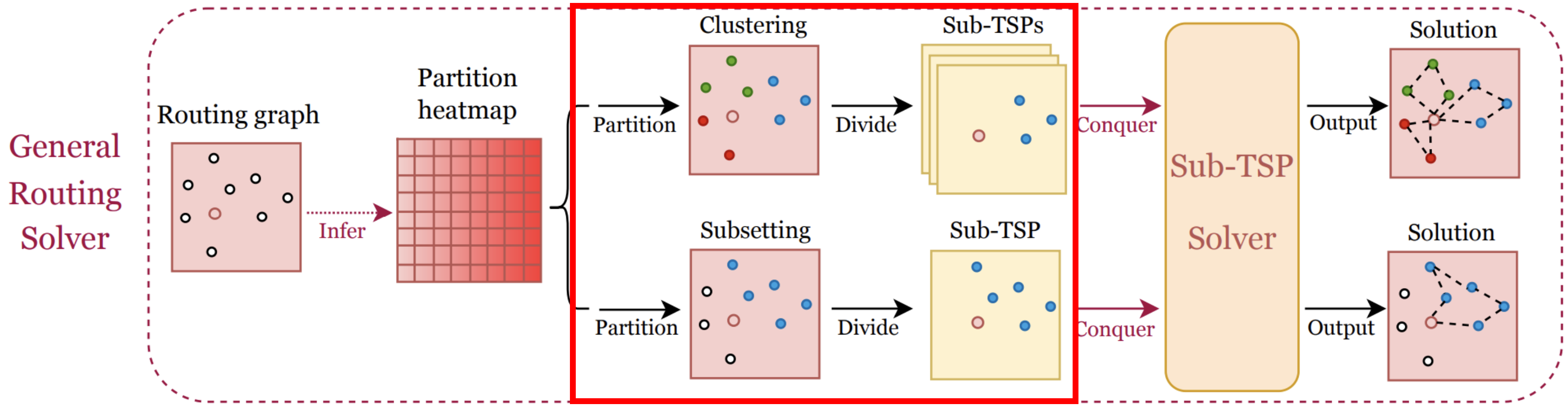
Output a partition heatmap through a parameterized GNN

$$\mathcal{H}_\phi(\rho) = [h_{ij}(\rho)]_{(n+1)\times(n+1)}$$

-$h_{i,j}$: *probability of node i and j belonging to the same subset*

-ρ: *input instance with* $n+1$ *node including* 0 *as the depot*

4. General Routing Solver - **Global policy as partition heatmap**

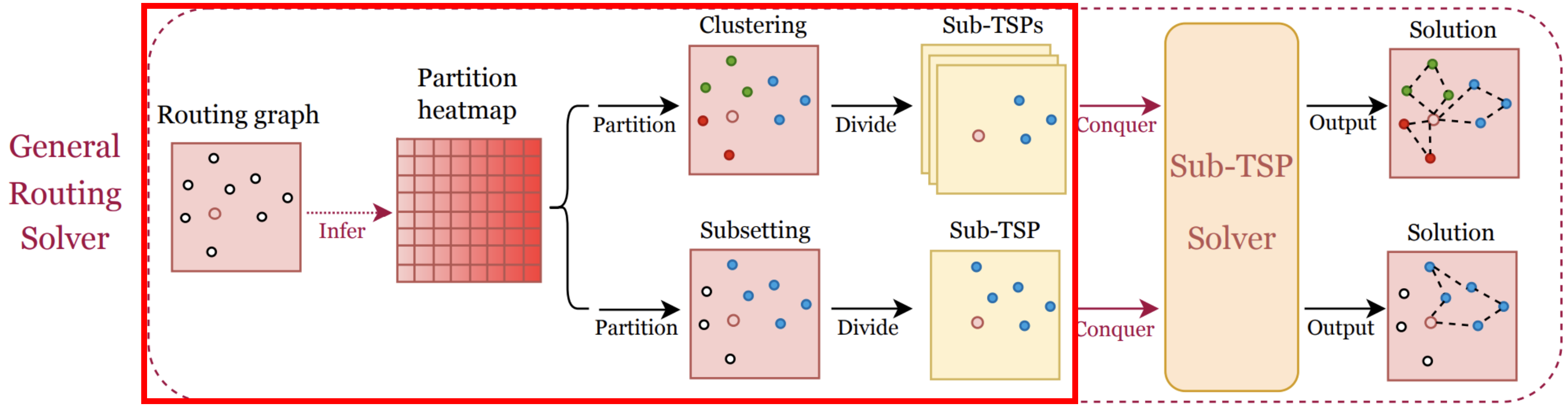


- Partition all nodes into multiple subsets for node clustering (CVRP, mTSP CARP..)
- Partition all nodes into two subsets for node subsetting (PCTSP, OP, CSP..)

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

- Θ: *problem specific constraint*

4. General Routing Solver - **Training algorithm**



- Infers partition heatmap $\mathcal{H}_{\phi}(\rho)$
- Samples node partitions in parallel
- Feeds the sampled partitions into GLOP(sub-TSP solver) for sub-TSP solutions
- Optimizes the expected final performance

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

  - $f_{TSP}: tour\ length\ of\ sub - TSP\ solution$
  - $GLOP_{\theta}: trained\ local\ policy\ generate\ sub - TSP\ solution$

6. Experiment

Method	TSP500			TSP1K			TSP10K		
	Obj.	Gap(%)	Time	Obj.	Gap(%)	Time	Obj.	Gap(%)	Time
Concorde (Applegate et al. 2006)	16.55	0.00	40.9m	23.12	0.00	8.2h	-	-	-
LKH-3 (Helsgaun 2017)	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 (Kool, van Hoof, and Welling 2019)	22.60	36.6	5.8m	42.53	84.0	22m	430	499	3.5m
LCP (Kim, Park, and Kim 2021)	20.82	25.8	29m	36.34	57.2	34m	357	397	4.3m
GCN+MCTS $\times 12$ (Fu, Qiu, and Zha 2021)	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 (Hottung, Kwon, and Tierney 2022)	24.04	45.3	1.0h	47.79	107	8.6h		OOM	
DIMES+S (Qiu, Sun, and Yang 2022)	19.06	15.0	2.0m	26.96	16.1	2.4m	86.25	20.0	3.1m
DIMES+MCTS $\times 12$ (Qiu, Sun, and Yang 2022)	17.01	2.78	1.0m+2.1m	23.86	3.20	2.6m+1.0m	76.02	5.90	13.7m+20m
Tspformer* (Yang et al. 2023)	17.57	5.97	3.1m	27.02	16.9	5.0m	-	-	-
H-TSP (Pan et al. 2023)	-	-	-	24.65	6.62	47s	77.75	7.32	48s
Pointerformer (Jin et al. 2023)	17.14	3.56	1.0m	24.80	7.30	6.5m	-	-	-
DeepACO (Ye et al. 2023)	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 $\times 12$		OOM	
POMO-EAS		OOM	
DIMES+MCTS $\times 12$		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 $\times 1$	1.10	7.5m
POMO-EAS $\{T = 60\}$	18.8	20m
DIMES+MCTS $\times 1$	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** (8.2s)
ATSP250	4.49 (12s)	**2.04** (9.3s)
ATSP1000	-	**2.33** (15s)

Table 4: Comparison results on ATSP. The results were updated in July 2024.

6. Experiment

Method	Time	Uniform Gap(%)	Expansion Gap(%)	Expansion Det.(%)	Explosion Gap(%)	Explosion Det.(%)	Implosion Gap(%)	Implosion 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 Obj.	CVRP1K Time (s)	CVRP2K Obj.	CVRP2K Time (s)	CVRP5K Obj.	CVRP5K Time (s)	CVRP7K Obj.	CVRP7K 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. 2022a)	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.

6. Experiment

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. 2022a)	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.

6. Experiment

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. 2021a)	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.

5. Conclusion and Limitation

Conclusion

- leverage scalability of the NAR paradigm and meticulousness of the AR paradigm
- demonstrate competitive and SOTA real-time performance on large scale TSP, ATSP, CVRP, PCTSP

Limitation

- 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