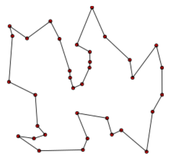
Unify ML4TSP: Drawing Methodological Principles for TSP and Beyond from Streamlined Design Space of Learning and Search

Yang Li[12], Jiale Ma[12], Wenzheng Pan[1], Runzhong Wang[13], Haoyu Geng[1], Nianzu Yang[1], Junchi Yan[12]

[1]Shanghai Jiao Tong University [2]Shanghai Innovation Institute [3]Massachusetts Institute of Technology

Background and Motivation

Combinatorial Optimization (CO): finding an optimal object from a finite set of objects, where the set of feasible solutions is discrete or can be reduced to a discrete set.



Travelling Salesman Problem

For example, TSP: Given a complete weighted graph, find the shortest Hamiltonian cycle (a cycle that visits each vertex exactly once) in the graph.

BackGround:

- Learning and search elements are often interleaved with each other, and common designs among existing learning-based solvers may manifest in varying implementations, making their effects and contributions less transparent, let alone their intricate interplay.
- The lack of crisp analysis of the whole ML4CO system hinders the determination of the role of learning and the establishment of desirable principles for learning designs.

What does this paper do?

- **This study utilizes TSP as a major case study, with adaptations for other CO problems, dissecting established mainstream learning-based solvers to outline a comprehensive design space.**
- We present ML4TSPBench, which advances a unified modular streamline incorporating existing technologies in both learning and search for transparent ablation, aiming to **reassess the role of learning and discern which parts of existing techniques are genuinely beneficial** and which are not.
- This further leads to the investigation of **desirable principles of learning designs and high-level paradigms** and the exploration of concepts guiding method designs.
- Leveraging the findings, we **propose enhancements to existing methods** to compensate for their missing attributes, thereby advancing performance and enriching the library.
- The strategic decoupling and organic recompositions **yield a factory of new TSP solvers**, where we investigate synergies across various method combinations and pinpoint the optimal design choices to create more powerful ML4TSP solvers.
- It facilitates and offers a reference for future research and engineering endeavors.

Modular Framework for ML4TSP Solvers

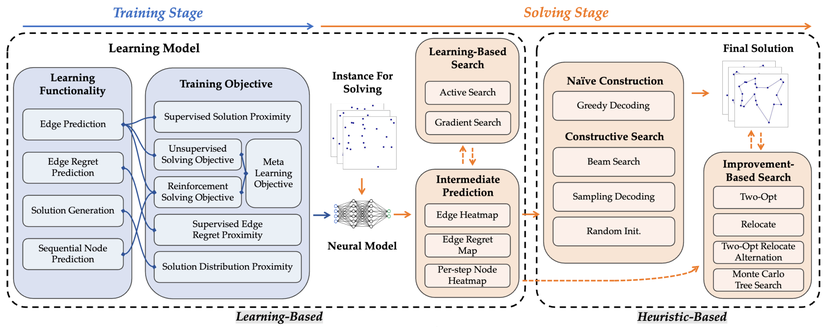


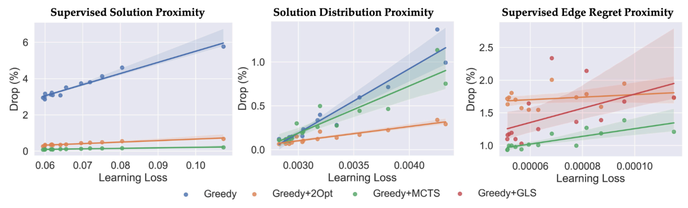
Figure 1: Overview of our proposed ML4TSPBench[1]. Blue and orange stand for training and testing processes, respectively. Dashed lines indicate the optional processes in pipelines.

- **Training:** Given instances and their (optional) labels e.g. reference solutions, the training stage involves different learning functionalities and training objectives to exploit the effective heuristics from data.
- **Solving:** Given instances, the solving stage follows a construction-refinement pipeline whereby a complete (feasible) solution is generated first then it is improved through the improvement-based search. The construction is based on the intermediate predictions produced by the trained model, and learning-based search is optionally applied to improve prediction quality by updating the parameters of the model or the intermediate predictions.

Empirical Investigations

- ***Q1.*** *How does learning benefit testing stage problem solving?*
- ***Q2.*** *What are the desirable design principles for ML4TSP methods?*
- ***Q3.*** *Which combination can enhance solving performance?*

Q1. Learning Generally Improves Solving with Compatible Search Algorithms



- Overall, there exists a positive correlation between learning quality and problem-solving performance. However, the introduction of a more robust search mechanism tends to mitigate the advantages of neural predictions.
- Once the learning loss surpasses a certain threshold (e.g., less than 0.003 for the distribution objective), the efficacy of translating improvements in neural predictions into enhancements in problem-solving quality may diminish.

Q2. Desirable Design Principles that Benefits ML4TSP Solving

• Joint Probability Estimation Helps Capture Variable Correlations

Table 1: Joint probability estimation ablation based on methods with *Edge Pred* as the learning functionality. 8× Greedy: sample 8 heatmaps in parallel then enforce greedy.

Objective	Joint Prob.	Solving Stage		TSP-50		TSP-100		TSP-500	
		Construction	Impr. Search	Drop↓	Time	Drop↓	Time	Drop↓	Time
SL	✗	Greedy	–	1.530%	7.6ms	7.078%	10.6ms	12.237%	0.1s
SL + Node Norm	✓	Greedy	–	0.674%	7.4ms	2.721%	10.1ms	8.496%	0.1s
Generative	✓	Greedy	–	**0.076%**	0.229s	**0.186%**	0.4s	**6.695%**	1.6s
SL	✗	Greedy	2-Opt	0.115%	7.6ms	0.649%	12.0ms	2.089%	0.1s
SL + Node Norm	✓	Greedy	2-Opt	0.082%	7.4ms	0.292%	11.7ms	1.325%	0.1s
Generative	✓	Greedy	2-Opt	**0.046%**	0.2s	**0.077%**	0.4s	**0.883%**	1.6s
SL	✗	Beam (n=1280)	–	0.023%	1.1s	0.673%	2.3s	23.629%	11.6s
SL + Node Norm	✓	Beam (n=1280)	–	0.016%	1.1s	0.545%	2.2s	21.678%	13.7s
Generative	✓	8× Greedy	–	**0.007%**	0.7s	**0.011%**	2.5s	**1.443%**	10.3s
SL	✗	Beam (n=1280)	2-Opt	0.015%	1.1s	0.149%	2.3s	2.431%	12.9s
SL + Node Norm	✓	Beam (n=1280)	2-Opt	0.012%	1.1s	0.156%	2.3s	2.189%	14.0s
Generative	✓	8× Greedy	2-Opt	**0.006%**	0.7s	**0.009%**	2.5s	**0.320%**	10.4s

- Proposed Node-wise normalization $\mathbf{F}'_{i,j} = \frac{\exp(\mathbf{F}_{i,j})}{\sum_{k=i}^{n}\exp(\mathbf{F}_{i,k})} \cdot \begin{bmatrix} n-2 & 0 \\ 0 & 2 \end{bmatrix}$

• Symmetry Solution Representation Introduces Equivalence Awareness

Table 2: Symmetry ablation based on methods with *Generative* as training objective.

Symmetry	Solving Stage		TSP-500	
	Construction	Impr. Search	Drop↓	Time
✗	Greedy	–	9.639%	1.2s
✓	Greedy	–	**6.695%**	1.6s
✗	Greedy	2-Opt	1.524%	1.2s
✓	Greedy	2-Opt	**0.883%**	1.6s
✗	8× Greedy	–	4.613%	6.9s
✓	8× Greedy	–	**1.443%**	10.3s
✗	8× Greedy	2-Opt	0.686%	6.9s
✓	8× Greedy	2-Opt	**0.320%**	10.4s

Table 3: Symmetry ablation based on methods with *RL* as the training objective. AS: active search.

Functionality	Symmetry	Solving	TSP-50		TSP-100	
			Drop↓	Time	Drop↓	Time
Edge Pred	✗	Sampling	5.238%	11.4ms	9.364%	94.5ms
+ Rand. Start	✓	Sampling	**2.157%**	72.7ms	**4.885%**	0.1s
Edge Pred	✗	AS+Sampling	3.086%	3.2s	4.234%	6.7s
+ Rand. Start	✓	AS+Sampling	**0.823%**	4.9s	**2.089%**	9.3s
Sequential	✗	Greedy	1.040%	0.1ms	2.520%	0.2ms
+ MultiStart	✓	Greedy	**0.179%**	1.7ms	**1.641%**	1.6ms
+ Sym. Baseline	✓	Greedy	0.877%	0.6ms	2.209%	0.6ms
Sequential	✗	Sampling	1.212%	0.1ms	2.846%	0.2ms
+ MultiStart	✓	Sampling	**0.156%**	1.3ms	**1.618%**	1.6ms
+ Sym. Baseline	✓	Sampling	1.041%	0.6ms	2.593%	0.6ms

• Online Optimization Complements Generalization Capability

• Search Friendliness Contributes to Learning-Search Synergy

Q3. Streamlining the ML4TSP Design Space Discovers More Powerful Solvers

• Evaluation of Heuristic Search with Proposed MCTS Variants

Table 6: Heuristic search ablation based on *Edge Pred* functionality and *SL* training objective.

Solving Stage		TSP-50		TSP-100		TSP-500	
Construction	Impr. Search	Drop↓	Time	Drop↓	Time	Drop↓	Time
Greedy	–	1.530%	7.6ms	7.078%	10.6ms	12.237%	0.1s
Beam (n=1280)	–	0.023%	1.1s	0.673%	2.3s	23.629%	11.6s
MCTS Solver (10t)		0.006%	56.7ms	0.006%	0.217s	0.433%	10.1s
Greedy	Two-Opt	0.115%	7.6ms	0.649%	12.0ms	2.089%	0.1s
Greedy	Guided LS (t=1s)	0.614%	1.1s	3.319%	1.2s	7.682%	45.5s
Greedy	MCTS	0.018%	8.4ms	0.154%	16.8ms	0.838%	1.3s
Beam (n=1280)	Two-Opt	0.015%	1.1s	0.149%	2.3s	2.431%	12.9s
Beam (n=50)	MCTS	0.005%	0.1s	0.009%	0.5s	**0.235%**	56.3s
Sampling (n=50)	MCTS	0.002%	0.1s	0.001%	0.5s	0.396%	58.1s
Random (n=10)	MCTS	0.004%	17.5ms	0.011%	81.4ms	0.519%	11.2s
Random (n=50)	MCTS	**0.002%**	62.5ms	**0.001%**	0.4s	0.301%	55.2s

• Recomposing to the Optimal

Table 7: Optimal recomposition of learning and search techniques. *: reference for drop estimation.

Training Stage		Solving Stage			TSP-50		TSP-100		TSP-500	
Learning Functionality	Training Objective	Learning Search	Construction	Impr. Search	Drop (%)↓	Time	Drop (%)↓	Time	Drop (%)↓	Time
–	–	–	PyConcorde (Applegate et al., 2006)		0.000±0.000*	73.6ms	0.000±0.000*	0.4s	0.000±0.000*	18.7s
–	–	–	LKH3 (trials=500) (Helsgaun, 2017)		0.000±0.000	0.2s	0.002±0.019	0.2s	0.322±0.174	0.7s
–	–	–	LKH3 (trials=5k) (Helsgaun, 2017)		–	–	0.000±0.006	1.6s	0.072±0.078	4.2s
–	–	–	LKH3 (trials=50k) (Helsgaun, 2017)		–	–	–	–	0.014±0.029	26.3s
Edge Pred	SL + Node Norm	–	Greedy	MCTS	0.021±0.114	7.4ms	0.080±0.261	16.3ms	0.578±0.420	1.3s
Edge Pred	SL	–	Random (n=50)	MCTS	0.002±0.025	62.5ms	**0.001±0.012**	0.4s	0.301±0.153	55.2s
Edge Pred.	SL + Node Norm	–	Random (n=50)	MCTS	0.004±0.026	50.4ms	0.002±0.018	0.4s	**0.142±0.106**	55.3s
Regret Pred	SL Regret	–	Greedy	Guided LS (t=10s)	0.004±0.035	10.1s	0.483±0.901	10.4s	–	–
Regret Pred	SL Regret	–	MCTS Solver (10t)		0.001±0.022	63.4ms	1.116±0.781	0.2s	–	–
Regret Pred	SL Regret	–	Random (n=50)	MCTS	**0.000±0.000**	0.2s	0.179±0.241	1.8s	–	–
Generation	Generative + Sym.	–	Greedy	Two-Opt	0.046±0.140	0.2s	0.077±0.160	0.4s	0.883±0.543	1.6s
Generation	Generative + Sym.	Gradient Search	Greedy	Two-Opt	0.012±0.053	0.7s	0.023±0.068	0.8s	0.445±0.253	3.8s
Generation	Generative + Sym.	Gradient Search	8× Greedy	Two-Opt	0.002±0.024	1.3s	0.003±0.022	5.0s	0.187±0.138	28.6s
Sequential	RL	–	Sampling + Multistart		0.156±0.265	62.5ms	1.618±0.893	1.6ms	–	–
Sequential	RL + Sym. Baseline	–	Greedy + Multistart	Two-Opt	0.122±0.244	0.1s	0.796±0.548	0.5s	–	–
Sequential	RL + Sym. Baseline	–	Sampling + Multistart	Two-Opt	0.110±0.242	0.2s	0.729±0.533	1.5s	–	–