LMask: Learn to Solve Constrained Routing Problems with Lazy Masking

Haijun Zou

Academy of Mathematics and System Science, CAS
University of Chinese Academy of Sciences

Joint work with Tianyou Li, Jiayuan Wu, Zaiwen Wen

Outline

Routing problems

- Routing problems are canonical combinatorial optimization tasks with wide-ranging applications in logistics, transportation, and supply chain management.
- The travelling salesman problem (TSP) is one of the most intensively studied problems in optimization, to find the shortest possible route that visits each city exactly once.
- The vehicle routing problem (VRP) extends TSP by introducing multiple vehicles and additional real-world constraints.

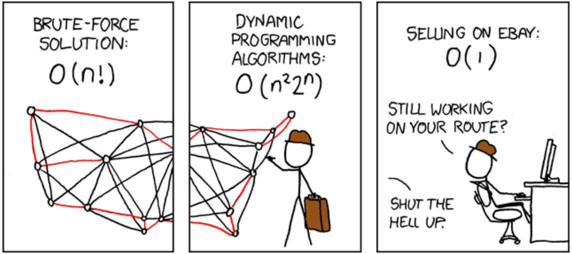


Figure: The long-standing difficult problem in combinatorial optimization.

Existing methods

- **Mixed Integer Linear Programming (MILP)**
  - Miller-Tucker-Zemlin (MTZ) formulation.
  - Dantzig-Fulkerson-Johnson (DFJ) formulation.

  Since these massive subtour elimination constraints make the model overly complex, the lazy-cut technique dynamically adds constraints during the cutting plane method.
- **Heuristic**
  - Simulated Annealing (SA) algorithm
  - Hybrid Genetic Search (HGS) algorithm
  - Lin-Kernighan-Helsgaun (LKH) algorithm
  - ...
- **Machine Learning approaches**
  - Neural constructive solvers: learn to construct solutions in an end-to-end manner
  - Neural iterative solvers: learn to iteratively refine a solution

Neural constructive solvers

- Independent for each specific routing problem with fixed scale.
  - Pointer Network is first introduced to solve TSPs in an auto-regressive model based on Recurrent Neural Networks and supervised learning (Vinyals et al., 2015).
  - Attention Model (AM) first adopts a transformer-based model to solve routing problems under a reinforcement learning framework (Kool et al, 2018).
  - Policy Optimization with Multiple Optima (POMO) significantly improved AM with diverse rollouts and data augmentations (Kwon et al., 2020).
  - Light Encoder and Heavy Decoder (LEHD) model designs a heavy-decoder transformer for stronger generalization to large-scale instances sizes (Luo et al., 2024).
- Foundation model for a range of VRP variants.
  - MVMoE is a multi-task vehicle routing solver utilizing a mixture-of-experts approach, capable of addressing 16 VRP variants with a single model (Zhou et al, 2024).
  - RouteFinder utilizes a unified VRP environment capable of efficiently handling any attribute combination with experiments on 48 VRP variants (Berto et al., 2024).

Mask preserves the feasibility of the route

- Generates a route in a node-by-node manner:

$$p_\theta(\pi;\mathcal{P}) = \prod_{t=1}^{T-1} p_\theta(\pi_{t+1}|\pi_{1:t};\mathcal{P}).$$

- Masking mechanism is applied in the transformer decoder to know which nodes are infeasible.

$$u_{(c)j} = \begin{cases} C \cdot \tanh\left(\frac{q_{(c)}^T k_j}{\sqrt{d_k}}\right), & \text{if } j \neq \pi_{t'},\ \forall t' < t \\ -\infty, & \text{otherwise.} \end{cases}$$

- Compute the final output probability vector p using a softmax:

$$p_\theta(\pi_t = i|\pi_{1:t-1};\mathcal{P}) = \frac{e^{u_{(c)i}}}{\sum_j e^{u_{(c)j}}}.$$

Failure in TSPTW

- The success of the masking mechanism in routing problems relies on
  - after applying a series of construction steps, the remaining subproblem becomes a smaller feasible instance of the original CO problem;
  - ground truth masks are easily obtainable for each step.
- However, such assumptions may fail in some routing problems, such as travelling salesman problem with time windows (TSPTW).
- Once a node is selected, the decision becomes irreversible, potentially leading to infeasible situations after several steps.

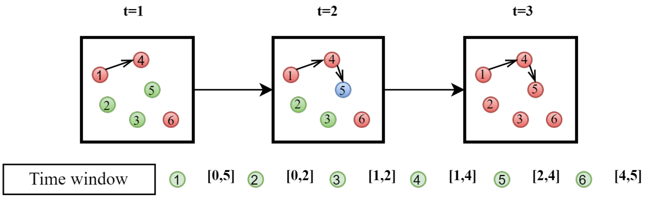


Figure: No node can be selected to satisfy the time windows.

Feasibility awareness

To deal with hard-constrained routing problems, neural constructive solvers are developed with feasibility awareness on the decoding process.

- Chen et al. develop a multi-step look-ahead method tailored for the TSPTW, incorporating problem-specific features and a large supervised learning dataset.
- Bi et al. propose a Proactive Infeasibility Prevention (PIP) framework based on preventative infeasibility masking, learnable decoders, and adaptive strategies to advance neural methods without the need for labeled training data.
- Our framework, LMask, utilizes a dynamic masking mechanism to solve constrained routing problems effectively and achieves SOTA feasibility rates and solution quality compared to existing neural methods.

Outline

A unified formulation of routing problems

- In the context of end-to-end learning, we typically do not use MILP formulation due to its high computational complexity and poor constraint satisfiability.
- Let $V = \{0, 1, \ldots, n\}$ denote the set of nodes and $\Pi := V^T$ represent the sequence space containing all possible routes of length T.
- A wide range of routing problems can be expressed as

$$\begin{aligned} \min_{\pi \in \Pi} \quad & f(\pi; \mathcal{P}) \\ \text{s.t.} \quad & c(\pi; \mathcal{P}) \leq 0, \\ & d(\pi; \mathcal{P}) = 0, \end{aligned} \tag{1}$$

  where $\mathcal{P}$ represents the problem instance.
- For example, $c(\pi; \mathcal{P}) \leq 0$ can represent time window constraints, draft limit constraints, and $d(\pi; \mathcal{P}) = 0$ can be the visit constraints that each node is exactly visited once.

Example: TSPTW and TSPDL

- TSP with time windows

$$
\begin{aligned}
\min \quad & f(\pi;\mathcal{P}) = \sum_{i=1}^{n} \|x_{\pi_i} - x_{\pi_{i+1}}\| + \|x_{\pi_{n+1}} - x_{\pi_1}\| \\
\text{s.t.} \quad & c_i(\pi;\mathcal{P}) = \sum_{t=1}^{i} t_{\pi_t,\pi_{t+1}} - l_{\pi_{i+1}} \leq 0, \quad i = 1,\dots,n, \\
& c_{n+i}(\pi;\mathcal{P}) = e_{\pi_{i+1}} - \sum_{t=1}^{i} t_{\pi_t,\pi_{t+1}} \leq 0, \quad i = 1,\dots,n, \\
& d_i(\pi;\mathcal{P}) = \sum_{t=1}^{n} \mathbb{1}_{\pi_t = i} - 1 = 0, \quad i = 1,\dots,n.
\end{aligned}
$$

- TSP with draft limits

$$
\begin{aligned}
\min \quad & f(\pi;\mathcal{P}) = \sum_{i=1}^{n} \|x_{\pi_i} - x_{\pi_{i+1}}\| + \|x_{\pi_{n+1}} - x_{\pi_1}\| \\
\text{s.t.} \quad & c_i(\pi;\mathcal{P}) = \sum_{t=1}^{i} q_{\pi_{t+1}} - D_{\pi_{i+1}} \leq 0, \quad i = 1,\dots,n, \\
& d_i(\pi;\mathcal{P}) = \sum_{t=1}^{n} \mathbb{1}_{\pi_t = i} - 1 = 0, \quad i = 1,\dots,n.
\end{aligned}
$$

Gibbs distribution

- Let Π^* be the optimal set of the problem (1) and $f^*(\mathcal{P})$ be the optimal objective value.
- One can represent the search for the optimal points by finding the target distribution:

$$q^*(\pi;\mathcal{P}) = \frac{1}{|\Pi^*|}\mathbb{1}_{\Pi^*}(\pi) = \begin{cases} \frac{1}{|\Pi^*|}, & \pi \in \Pi^*, \\ 0, & \pi \notin \Pi^*. \end{cases}$$

- A family of constrained Gibbs distributions can approximate the target distribution:

$$q_\lambda(\pi;\mathcal{P}) = \frac{1}{Z_\lambda}\exp\left(-\frac{f(\pi;\mathcal{P}) - f^*(\mathcal{P})}{\lambda}\right)\mathbb{1}_C(\pi) \xrightarrow{\lambda\to 0} q^*(\pi;\mathcal{P}), \quad \forall \pi \in \Pi,$$

  where $C := \{\pi \in \Pi : c(\pi;\mathcal{P}) \le 0, d(\pi;\mathcal{P}) = 0\}$ is the feasible set of the problem (1) and $Z_\lambda = \sum_{\pi\in C}\exp\left(-(f(\pi;\mathcal{P}) - f^*(\mathcal{P}))/\lambda\right)$.

Parameterization

- Simulated Annealing (SA) directly sample from the Gibbs distributions q_λ with shortcomings of slow convergence and difficulties in handling constraints.
- Constructing a parameterized distribution p_θ is often considered a more efficient method to sample a feasible route in the higher dimension.
- To reduce the discrepancy between the parameterized distribution p_θ and the Gibbs distribution q_λ, we minimize their KL divergence

$$\mathrm{KL}(p_\theta || q_\lambda) = \mathbb{E}_{p_\theta}[\log p_\theta] + \frac{1}{\lambda}\mathbb{E}_{p_\theta}[f(\pi; \mathcal{P})] - f^*(\mathcal{P}) + \log Z_\lambda,$$

  where the support of p_θ should be contained in the support of q_λ, namely C.
- Since $-\log Z_\lambda + f^*(\mathcal{P})$ is a constant with respect to θ, the loss function is

$$L(\theta; \mathcal{P}) := \mathbb{E}_{p_\theta(\cdot;\mathcal{P})}[f(\pi; \mathcal{P})] + \lambda \mathbb{E}_{p_\theta(\cdot;\mathcal{P})}[\log p_\theta(\pi; \mathcal{P})].$$

Outline

Masking mechanism in the auto-regressive model

- An auto-regressive distribution can be constructed for parameterization

$$p_\theta(\pi; \mathcal{P}) = \prod_{t=1}^{T-1} p_\theta(\pi_{t+1} | \pi_{1:t}; \mathcal{P}).$$

- $p_\theta(\pi_{t+1} | \pi_{1:t}; \mathcal{P})$ should explicitly exclude entirely infeasible points.
- To formalize this, we introduce the potential set $S(\pi_{1:t})$, defined as:

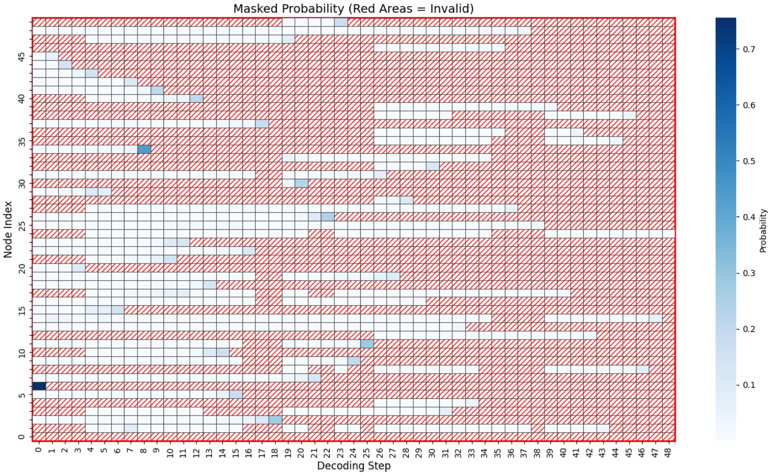
$$S(\pi_{1:t}) := \{\pi_{t+1} : \exists \pi_{t+2:T} \in V^{T-t-1}, [\pi_{1:t+1}, \pi_{t+2:T}] \in C\},$$

  which further induces the mask function $M(\pi_{t+1} | \pi_{1:t}; \mathcal{P}) := \mathbb{1}_{S(\pi_{1:t})}(\pi_{t+1})$.
- The conditional probability parameterized by the neural network takes the form:

$$p_\theta(\pi_{t+1} | \pi_{1:t}; \mathcal{P}) = \frac{e^{\phi_\theta(\pi_{t+1} | \pi_{1:t}; \mathcal{P})} M(\pi_{t+1} | \pi_{1:t}; \mathcal{P})}{\sum_{k=0}^{n} e^{\phi_\theta(k | \pi_{1:t}; \mathcal{P})} M(k | \pi_{1:t}; \mathcal{P})},$$

  where $\phi_\theta(\cdot | \pi_{1:t}; \mathcal{P})$ is produced by all intermediate layers.

Mask Visualization



Overestimation

- The pontential set of the vanilla TSP and CVRP is exactly accessible, leading to the success in the most of existing neural constructive methods.
- However, $S(\pi_{1:t})$ is sometimes computationally inaccessible for complex constraints since it may require an exhaustive lookahead until a complete solution is constructed.
- To address this, we propose the LazyMask algorithm which works with an overestimation set $\hat{S}(\pi_{1:t})$ representing the currently known complementary set of actions that are deemed impossible.
- LazyMask adaptively manages this set by establishing it via informed initialization strategies and incrementally refining it through backtracking.
- Since the potential set $S(\pi_{1:t})$ is a proper subset of its estimation $\hat{S}(\pi_{1:t})$, the estimation accuracy is enhanced through initialization and backtracking refinement.

Backtracking

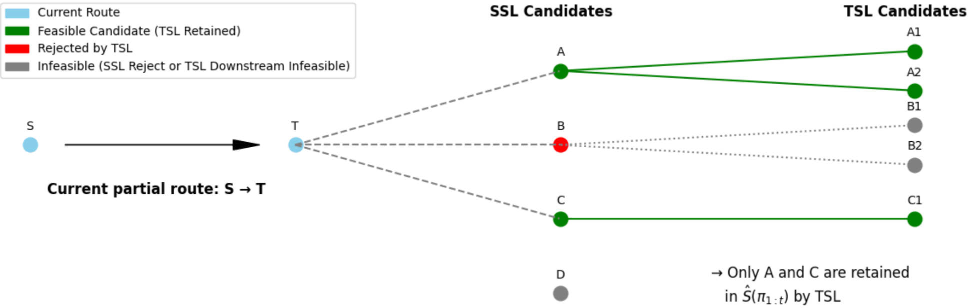
1. **Initialization**: We first initialize the overestimation set $\hat{S}$ to closely approximate the potential set S. The overestimation excludes nodes by basic feasibility checks, e.g., already visited nodes. Additional problem-specific rules are further applied to prune more complex infeasible actions.
2. **Feasibility check**: After selecting action π_t at construction step t, check if it is still possible to complete a full feasible solution. If $\hat{S}(\pi_{1:t})$ is empty, π_t is deemed invalid and then we execute **Backtracking**. Otherwise, we continue **Construction**.
3. **Backtracking**: Return to the solution construction step $t-1$ and remove the invalid action from the estimated feasible set:

$$\hat{S}(\pi_{1:t-1}) \leftarrow \hat{S}(\pi_{1:t-1}) \backslash \{\pi_t\}.$$

4. **Construction**: Sample the next action $\pi_{t+1} \sim p_\theta(\cdot|\pi_{1:t}; \mathcal{P})$.
5. **Repetition**: Repeat **Steps 2-4** until a full feasible route is found.

Two-step lookahead initialization

- The initialization of $\hat{S}(\pi_{1:t})$ significantly impacts the algorithm efficiency.
- **Single-step lookahead (SSL).** SSL examines unvisited nodes to verify whether they satisfy problem-specific constraints given the current route.
- **Two-step lookahead (TSL).** TSL performs one more lookahead step to exclude nodes that may seem feasible under SSL but lead to infeasible routes.



LazyMask algorithm

Algorithm 1 LazyMask algorithm

Input: routing problem instance $\mathcal{P}$, neural network p_θ, backtracking budget R.
Initialize $\pi_1 := 0$, $t := 1$, $r := 0$ and the overestimation set $\hat{S}(\pi_1)$.
while $t \leq T - 1$ **do**
    if $\hat{S}(\pi_{1:t}) = \emptyset$ and $r \leq R$ **then**
        Update $\hat{S}(\pi_{1:t-1}) := \hat{S}(\pi_{1:t-1}) \backslash \{\pi_t\}$.
        Set $t := t - 1$, $r := r + 1$.
    else
        if $\hat{S}(\pi_{1:t}) = \emptyset$ **then**
            $\hat{S}(\pi_{1:t}) := \{0, 1 \ldots, n\} \setminus \{\pi_1, \ldots, \pi_t\}$.
        end if
        Calculate the probability $p_\theta(\cdot|\pi_{1:t}; \mathcal{P})$ using $\hat{S}(\pi_{1:t})$.
        Sample $\pi_{t+1} \sim p_\theta(\cdot|\pi_{1:t}; \mathcal{P})$.
        Set $t := t + 1$ and initialize $\hat{S}(\pi_{1:t})$.
    end if
end while
Output: Route π.

Refinement intensity embedding

- Standard dynamic features in existing auto-regressive models, such as AM and POMO, implicitly assumes a one-pass forward construction.
- When backtracking occurs, this design renders the model state invariant to its search trace, leading to representation ambiguities that can hinder the model's learning ability.
- We propose the refinement intensity embedding (RIE) designed to enrich the decoder's input with essential context about the search trace.
    - Locally, RIE records how many times the current $\hat{S}(\pi_{1:t})$ has been refined. This count is represented as a capped one-hot vector of length N.
    - Globally, it signals whether the total number of backtracks has reached the the backtracking budget R, encoded as a 2-dimensional one-hot vector.

Training

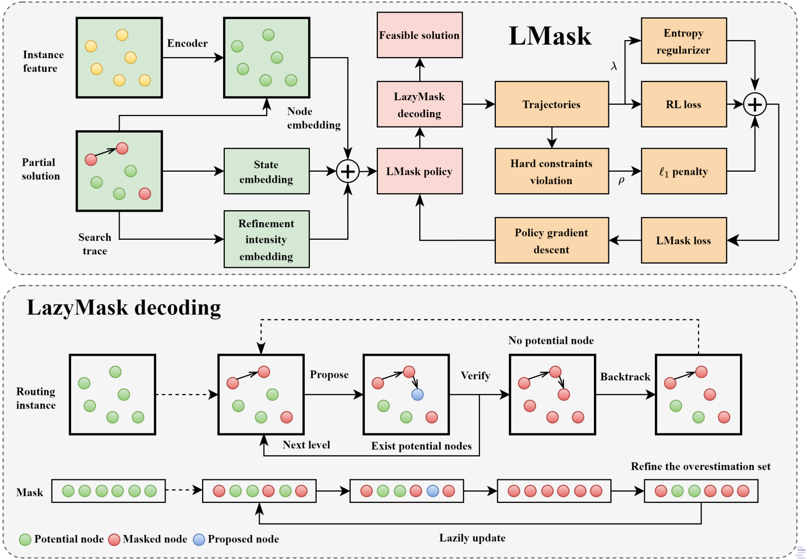
- During the early stages of training, the cost of backtracking is unaffordable because the policy distribution is not well-trained.
- Hence, we limit the backtracking budget R to balance computational efficiency and solution feasibility.
- Since the generated solution is not guaranteed feasible, we introduce an ℓ_1 penalty term for complex constraints while maintaining simpler constraints such as flow constraints.

$$\Psi_\rho(\pi;\mathcal{P}) := f(\pi;\mathcal{P}) + \rho \sum_{j=1}^{J} \max\left(c_j(\pi;\mathcal{P}), 0\right).$$

  where $\rho > 0$ is a given penalty parameter and $c_j(\pi;\mathcal{P})$, $j = 1, \ldots, J$ represent complex constraints in the constraint vector $c(\pi;\mathcal{P})$.
- The training loss is formally expressed as:

$$\min \quad \mathbb{E}_{\pi\sim p_\theta(\cdot;\mathcal{P})}\left[\Psi_\rho(\pi;\mathcal{P}) + \lambda \log p_\theta(\pi;\mathcal{P})\right].$$

LMask framework



Outline

Validity of LazyMask algorithm

- To ensure the effectiveness of Algorithm 1, we first prove that it always generates feasible solutions and has a non-zero probability of generating all feasible solutions.

Proposition

Suppose that the problem (1) *is feasible, and that the backtracking budget in Algorithm 1 is set to* $R = +\infty$. *Then,*

1. *any solution* π *generated by Algorithm 1 is feasible;*
2. *Algorithm 1 assigns a non-zero probability to generate any feasible solution* π.

- It is critical to show that infeasible solutions are not allowed.
- It is also important to demonstrate that no feasible solution is excluded by Algorithm 1.

Validity of Probabilistic Model

- There is a gap between the original routing problem and its probabilistic model.
- We first give the assumption of the approximation error, which ensures the efficient parameterization of the auto-regressive neural network p_θ.

Assumption

We assume that the auto-regressive neural network p_θ has sufficient expressive power to approximate the target distribution q_λ. Specifically, the approximation error $\delta(\lambda)$ defined as follows satisfies:

$$\delta(\lambda) = \min_{\theta} \max_{\mathcal{P} \in \mathcal{D}} \mathrm{KL}(p_\theta(\cdot; \mathcal{P}) \parallel q_\lambda(\cdot; \mathcal{P})) \leq c/\lambda, \ \lambda > 0,$$

where c is a small constant. Let $p_{\theta^*(\lambda)}$ be the corresponding optimal distribution.

Validity of Probabilistic Model

- The following Theorem establishes an upper bound on the probability that a sampled solution from the learned distribution deviates from the optimal objective value.

Theorem

Suppose that Assumption 1 holds. We define $\Delta(\mathcal{P}) := \min_{\pi \in C \setminus \Pi^*} f(\pi; \mathcal{P}) - f^*(\mathcal{P})$*. Then, for any* $\epsilon > 0$ *and* $\Delta(\mathcal{P}) \geq \lambda > 0$*, the following inequality holds:*

$$\mathbb{P}_{p_{\theta^*(\lambda)}}(f(\pi;\mathcal{P}) \geq f^*(\mathcal{P}) + \epsilon) \leq \frac{|C|\,\Delta(\mathcal{P}) e^{-\Delta(\mathcal{P})/\lambda}}{|\Pi^*| \max\{\epsilon, \Delta(\mathcal{P})\}} + \sqrt{\frac{c}{2\lambda}}.$$

- This intrinsic trade-off of the entropy regularization coefficient λ underscores the interplay between concentration, exploration, and approximation quality.

Outline

Experimental settings

- **Baseline.**
  - For traditional solvers, we adopt PyVRP,ORTools and LKH, which are all highly efficient open-source solvers for routing problems.
  - We also test the heuristic greedy algorithms Greedy-L and Greedy-C.
  - For neural solvers, we compare our method against PIP and PIP-D, which proactively mask actions that could lead to future infeasibility.
- **Results.** We generate 10,000 instances for each of the three hardness levels—**easy**, **medium**, and **hard**—for the TSPTW and TSPDL.

Performance on TSPTW

Nodes		$n = 50$					$n = 100$				
Method		Infeasible Sol.	Infeasible Inst.	Obj.	Gap	Time	Infeasible Sol.	Infeasible Inst.	Obj.	Gap	Time
Easy	PyVRP	-	0.00%	7.31	*	1.7h	-	0.00%	10.19	*	4.3h
	LKH3	-	0.00%	7.31	0.00%	1.9h	-	0.00%	10.21	0.29%	7.2h
	OR-Tools	-	0.00%	7.32	0.21%	1.7h	-	0.00%	10.33	1.43%	4.3h
	Greedy-L	31.84%	31.84%	9.49	30.28%	$\ll$1s	33.23%	33.23%	13.92	36.80%	$\ll$1s
	Greedy-C	0.00%	0.00%	26.09	257.63%	$\ll$1s	0.00%	0.00%	52.11	411.95%	$\ll$1s
	PIP	0.28%	0.01%	7.51	2.70%	9s	0.16%	0.00%	10.57	3.57%	29s
	PIP-D	0.28%	0.00%	7.50	2.57%	10s	0.05%	0.00%	10.66	4.41%	31s
	LMask	**0.06%**	**0.00%**	**7.45**	**2.02%**	7s	**0.01%**	**0.00%**	**10.50**	**3.11%**	17s
Medium	PyVRP	-	0.00%	13.03	*	1.7h	-	0.00%	18.72	*	4.3h
	LKH3	-	0.00%	13.02	0.00%	2.9h	-	0.01%	18.74	0.16%	10.3h
	OR-Tools	-	15.12%	13.01	0.12%	1.5h	-	0.52%	18.98	1.40%	4.3h
	Greedy-L	76.98%	76.98%	15.05	17.02%	$\ll$1s	77.52%	77.52%	23.36	25.43%	$\ll$1s
	Greedy-C	42.26%	42.26%	25.40	96.45%	$\ll$1s	18.20%	18.20%	51.69	176.58%	$\ll$1s
	PIP	4.82%	1.07%	13.41	3.07%	10s	4.35%	0.39%	19.62	4.73%	29s
	PIP-D	4.14%	0.90%	13.46	3.45%	9s	3.46%	0.03%	19.80	5.70%	31s
	LMask	**0.06%**	**0.00%**	**13.25**	**1.73%**	9s	**0.10%**	**0.00%**	**19.55**	**4.46%**	20s
Hard	PyVRP	-	0.00%	25.61	*	1.7h	-	0.01%	51.27	0.00%	4.3h
	LKH3	-	0.52%	25.61	0.00%	2.3h	-	0.95%	51.27	0.00%	1d8h
	OR-Tools	-	65.11%	25.92	0.00%	0.6h	-	89.25%	51.72	0.00%	0.5h
	Greedy-L	70.94%	70.94%	26.03	0.29%	$\ll$1s	93.17%	93.17%	52.20	0.29%	$\ll$1s
	Greedy-C	53.47%	53.47%	26.36	1.43%	$\ll$1s	81.09%	81.09%	52.70	1.42%	$\ll$1s
	PIP	5.65%	2.85%	25.73	1.12%	9s	31.74%	16.68%	51.48	0.80%	28s
	PIP-D	6.44%	3.03%	25.75	1.20%	9s	13.60%	6.60%	51.43	0.68%	31s
	LMask	**0.00%**	**0.00%**	**25.71**	**0.10%**	6s	**0.00%**	**0.00%**	**51.38**	**0.21%**	18s

Performance on TSPDL

Nodes		$n = 50$					$n = 100$				
Method		Infeasible Sol.	Infeasible Inst.	Obj.	Gap	Time	Infeasible Sol.	Infeasible Inst.	Obj.	Gap	Time
Medium	LKH	-	0.00%	10.85	*	2.3h	-	0.00%	16.36	*	10.2h
	Greedy-L	99.87%	99.87%	15.34	65.93%	$\ll$1s	100.00%	100.00%	-	-	$\ll$1s
	Greedy-C	0.00%	0.00%	26.12	144.33%	$\ll$1s	0.00%	0.00%	52.14	222.79%	$\ll$1s
	PIP	1.75%	0.17%	11.23	5.09%	8s	2.50%	0.16%	17.68	9.39%	21s
	PIP-D	2.29%	0.22%	11.27	5.44%	8s	1.83%	0.23%	17.80	10.12%	23s
	LMask	**0.03%**	**0.01%**	**11.14**	**2.75%**	6s	**0.20%**	**0.05%**	**17.04**	**4.24%**	15s
Hard	LKH	-	0.00%	13.25	*	2.6h	0.00%	0.00%	20.76	*	15.8h
	Greedy-L	100.00%	100.00%	-	-	$\ll$1s	100.00%	100.00%	-	-	$\ll$1s
	Greedy-C	0.00%	0.00%	26.09	100.25%	$\ll$1s	0.00%	0.00%	52.16	155.38%	$\ll$1s
	PIP	4.83%	2.39%	13.63	4.49%	8s	29.34%	21.65%	22.35	9.71%	20s
	PIP-D	4.16%	0.82%	13.79	5.68%	8s	13.51%	8.43%	22.90	12.57%	23s
	LMask	**0.19%**	**0.04%**	**13.57**	**2.52%**	6s	**0.80%**	**0.26%**	**21.63**	**4.34%**	15s

Compared to traditional solvers, LMask has a significant advantage in algorithmic efficiency due to the inference capability of neural networks. Furthermore, LMask shows significantly better solution feasibility and gap compared to other neural methods.

Performance on TSPTW benchmark

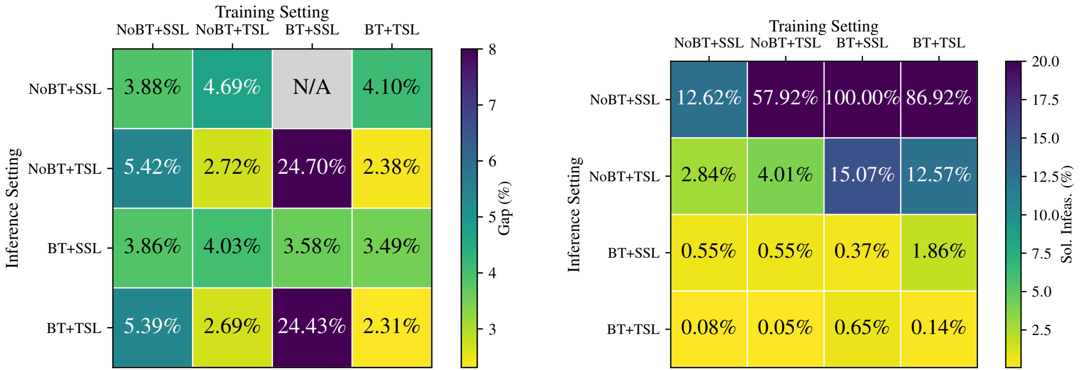
- Dumas et al. presented the well-known TSPTW benchmark dataset in 1995, which involves instances with various problem sizes and time window widths.
- We further evaluate all neural solvers on the TSPTW benchmark dataset.

Nodes	$n = 20$			$n = 40$			$n = 60$			$n = 80$		
Method	**Infeas.**	**Obj.**	**Gap**	**Infeas.**	**Obj.**	**Gap**	**Infeas.**	**Obj.**	**Gap**	**Infeas.**	**Obj.**	**Gap**
PIP	5%	337.00	5.2%	45%	428.09	4.6%	20%	580.25	11.5%	22.2%	644.43	8.7%
PIP-D	5%	336.63	5.2%	25%	460.27	6.3%	40%	608.67	13.1%	66.7%	662.67	12.0%
LMask	5%	332.74	3.9%	10%	450.44	3.7%	0%	543.50	4.4%	11.1%	625.25	5.1%

- Among all problem sizes, LMask has significantly lower infeasibility rates and optimal gaps, compared to PIP and PIP-D.

Ablation on backtracking

- We evaluate the effect of backtracking and initialization strategies by enumerating all 16 combinations of their usage at training and inference on medium TSPTW-50.

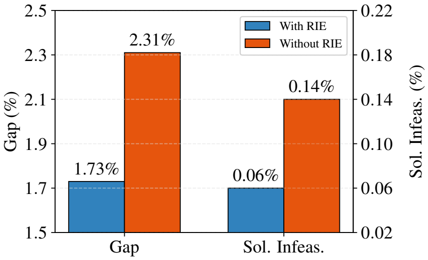


Ablation on RIE

- We compare the performance of models with and without RIE on medium TSPTW-50.

  - Excluding RIE leads to degraded performance for models employing LazyMask decoding.

  - Introducing RIE enables the model to adapt to dynamic search behaviors and enhances its robustness.



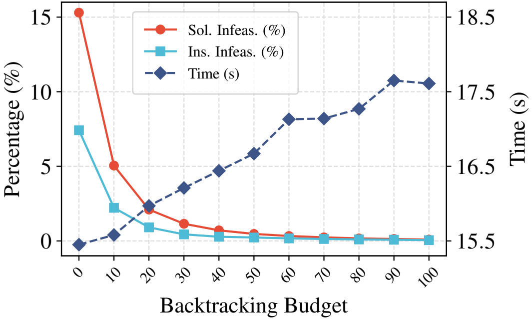
- Incorporating RIE leads to a substantial reduction in optimality gap and a moderate improvement in feasibility.

Ablation on backtracking budget

- We evaluate the impact of the backtracking budget R on the TSPTW100-hard dataset.

  - The inference time grows approximately linearly with R, exhibiting a modest slope.

  - Both infeasibility metrics decrease sharply initially, followed by a more gradual decline as R futher increases.



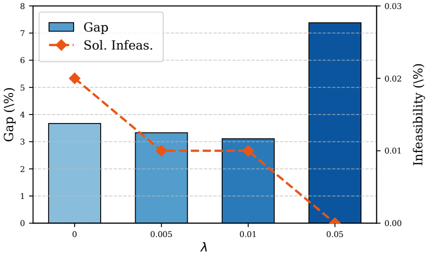
- It is shown that backtracking improves solution feasibility at a reasonable cost.

Effect of entropy term

- We report the evaluation results on the easy TSPTW-100 dataset for models trained with different entropy coefficients $\lambda = 0, 0.005, 0.01, 0.05$.

- Our theorem implies that with probability $1-\delta$, it holds

$$f(\pi;\mathcal{P}) \leq f^*(\mathcal{P}) + \frac{|C|\, e^{-\Delta(\mathcal{P})/\lambda}}{|\Pi^*|\,(\delta - \sqrt{\frac{c}{2\lambda}})}.$$



- The intrinsic trade-off between exploration and concentration in the probabilistic model.
- Choosing an appropriate entropy coefficient improves solution optimality and feasibility.

Conclusions

- We propose a novel framework, LMask, for solving hard-constrained routing problems by distinct mask mechanisms.
- We introduce the LazyMask algorithm to take advantage of the masking mechanism and dynamically decode feasible solutions with backtracking for general routing problems.
- The refinement intensity embedding is employed to encode the search trace into the model, mitigating representation ambiguities induced by backtracking.
- Theoretical guarantees are provided for validity and probabilistic optimality of LMask.
- Extensive experiments on TSPTW and TSPDL demonstrate that LMask achieves SOTA feasibility rates and solution quality, outperforming existing neural methods.

Many Thanks For Your Attention!