LMask: Learn to Solve Constrained Routing Problems with Lazy Masking

Haijun Zou

State Key Laboratory of Mathematical Sciences
Institute of Computational Mathematics and Scientific/Engineering Computing
Academy of Mathematics and Systems Science
Chinese Academy of Sciences, China

Joint work with Tianyou Li, Jiayuan Wu, Zaiwen Wen

SIAM Conference on Optimization (OP26)

June 4, 2026

Outline

Routing problems

- 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.
- TSP variants extend beyond the TSP by introducing additional real-world constraints, e.g., time windows (TSPTW), and draft limits (TSPDL).
- Vehicle routing problem (VRP) variants generalize TSP variants by optimizing a set of routes for a fleet of vehicles originating from a depot, rather than a single route.

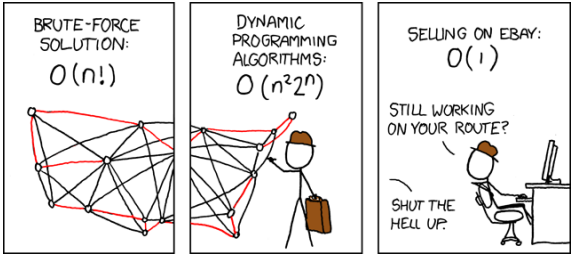


Figure: The long-standing difficult problem in combinatorial optimization.

Sequence-based formulation

- The MILP formulation suffers from high computational complexity and poor constraint satisfiability.
- Let $\Pi := V^T$ 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

- The sequence-based formulation provides a simple way to calculate quantities accumulated along the route.
- In TSPTW, the start of service time τ_{i+1} at node π_{i+1} for a given route π can be derived recursively: $\tau_{i+1} = \max\left(\tau_i + s_{\pi_i} + t_{\pi_i,\pi_{i+1}}, e_{\pi_{i+1}}\right)$, $i = 1, \ldots, n$, with $\tau_1 = 0$.
- A streamlined sequence-based formulation for TSPTW:

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

Neural constructive solvers for ordinary routing problems

- 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. 2023].
- 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].

Outline

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 $C := \{\pi \in \Pi : c(\pi;\mathcal{P}) \leq 0, d(\pi;\mathcal{P}) = 0\}$ 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

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

  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

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

Constrained auto-regressive model

- **Probabilistic factorization**
  - Since the solution space is a sequence space Π, the joint probability distribution is factorized via the chain rule:

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

  - A neural network is used to parameterize the conditional probabilities $p_\theta(\cdot \mid \pi_{1:t}; \mathcal{P})$.
- **Encoder-decoder network architecture**
  - **Encoder**: maps the problem instance $\mathcal{P}$ to node embeddings $H \in \mathbb{R}^{n \times d}$.
  - **Decoder**: generates the probability distribution over candidate nodes step-by-step based on the context.

Decoder with feasibility masks

- **Logits generation**
  - A query vector q_t is constructed based on the partial route $\pi_{1:t}$ and node embeddings.
  - Logits are computed via cross-attention with node embeddings:

$$\phi_\theta(\cdot \mid \pi_{1:t}; \mathcal{P}) = C_{\text{clip}} \cdot \tanh\left(\frac{q_t (HW)^\top}{\sqrt{d}}\right).$$

- **Probability generation**
  - To ensure feasibility, we first identify the potential set $S(\pi_{1:t})$:

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

  - The final probability is obtained by normalizing the masked logits:

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

Failure in TSPTW

- Masking effectiveness in ordinary routing problems relies on:
  - Feasibility is directly governed by node-wise accumulated quantities.
  - Remaining subproblems are guaranteed to remain feasible throughout construction.
- Ordinary VRPs often satisfy the latter by assuming an **unlimited vehicle fleet**.
- However, such an assumption fails to hold in TSP variants.
- Once a node is selected, the decision becomes irreversible, potentially leading to infeasible situations after several steps.

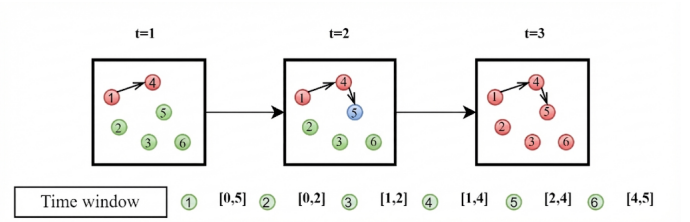


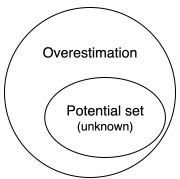
Figure: No node can be selected to satisfy the time windows.

Outline

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.

- Adaptively manages this set by establishing it via
  - informed initialization strategies
  - lazy refinement through backtracking.



Backtracking

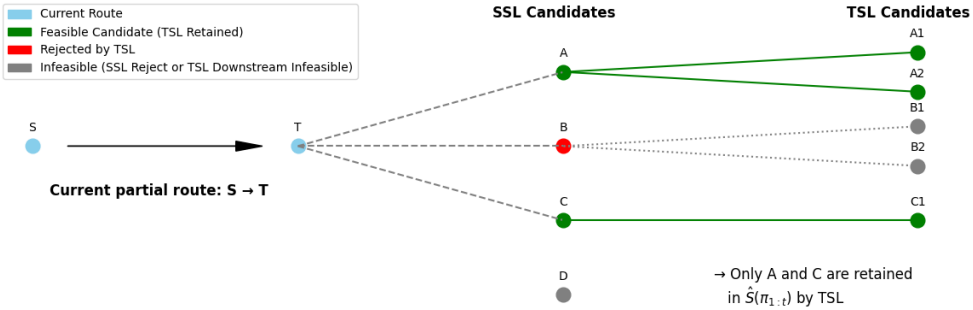
1. **Feasibility check**: Before extending the current partial route $\pi_{1:t}$, check if it is still possible to complete a full feasible solution. If $\hat{S}(\pi_{1:t})$ is empty, π_t is deemed invalid and then we execute **Backtracking**. Otherwise, we continue **Construction**.
2. **Backtracking**: Return to the solution construction step $t-1$ and remove the invalid action from the estimated feasible set:

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

3. **Construction**: Sample the next action $\pi_{t+1} \sim p_\theta(\cdot|\pi_{1:t}; \mathcal{P})$.
4. **Initialization**: After selecting action π_{t+1}, initialize the overestimation set $\hat{S}(\pi_{1:t+1})$ for the extened partial route. 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.
5. **Repetition**: Repeat the above steps until a full feasible route is found.

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 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 visit constraints.

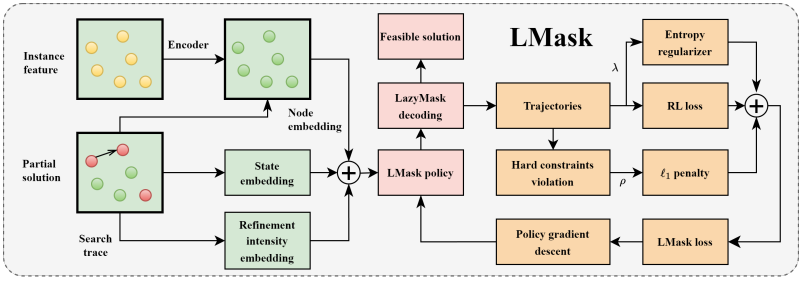
$$\Psi_\rho(\pi;\mathcal{P}) := f(\pi;\mathcal{P}) + \rho \sum_{j=1}^{J} \max\left(c_j(\pi;\mathcal{P}), 0\right).$$

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

- 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

We define $\Delta(\mathcal{P}) := \min_{\pi \in C \setminus \Pi^*} f(\pi; \mathcal{P}) - f^*(\mathcal{P})$. *Then, under a mild expressivity assumption, 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

- **Hardware:** `Intel Xeon Gold 6326` CPUs and `NVIDIA Tesla A800` GPUs.
- **Baselines:**
  - Traditional solvers: PyVRP, OR-Tools and LKH3.
  - Simple heuristics with backtracking: Random-L and Random-C.
  - Neural solvers: PIP and PIP-D (state-of-the-art for handling complex constraints).
- **Datasets:**
  - Synthetic TSPTW and TSPDL datasets (10,000 instances each, varied hardness).
  - A real-world TSPTW benchmark for measuring generalization abilities.
- **Metrics:**
  - **Infeasibility rates: Ins.** (failed instances) and **Sol.** (infeasible generated solutions).
  - **Gap:** Relative difference to PyVRP or LKH3 on shared feasible instances.
  - **Objective:** Average route length of the best feasible solutions.
  - **Time:** Total inference time for the dataset.

Performance on TSPTW synthetic datasets

	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
	Random-L	56.02%	0.04%	14.55	99.23%	1.3m	95.17%	9.37%	30.66	201.20%	6.1m
	Random-C	62.31%	0.03%	19.12	162.03%	1.4m	98.17%	27.30%	44.20	333.74%	6.1m
	PIP	0.28%	0.01%	7.51	2.73%	9s	0.16%	**0.00%**	10.57	3.78%	29s
	PIP-D	0.28%	**0.00%**	7.50	2.60%	10s	0.05%	**0.00%**	10.66	4.62%	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
	Random-L	98.31%	32.18%	18.91	47.04%	1.6m	100.00%	100.00%	-	-	5.8m
	Random-C	91.17%	8.18%	21.04	61.91%	1.6m	100.00%	100.00%	-	-	5.9m
	PIP	4.82%	1.07%	13.41	2.93%	10s	4.35%	0.39%	19.61	4.79%	29s
	PIP-D	4.14%	0.90%	13.46	3.31%	9s	3.46%	0.03%	19.80	5.76%	31s
	LMask	**0.04%**	**0.00%**	**13.25**	**1.68%**	6s	**0.05%**	**0.00%**	**19.51**	**4.23%**	18s
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
	Random-L	100.00%	100.00%	-	-	1.6m	100.00%	100.00%	-	-	5.6m
	Random-C	100.00%	99.82%	25.98	1.22%	1.6m	100.00%	100.00%	-	-	5.8m
	PIP	5.65%	2.85%	25.73	0.18%	9s	31.74%	16.68%	51.48	0.37%	28s
	PIP-D	6.44%	3.03%	25.75	0.27%	9s	13.59%	6.60%	51.43	0.32%	31s
	LMask	**0.00%**	**0.00%**	**25.71**	**0.10%**	6s	**0.00%**	**0.00%**	**51.38**	**0.21%**	18s

Performance on TSPDL synthetic datasets

	Nodes	$n = 50$					$n = 100$				
	Method	Infeasible Sol.	Infeasible Inst.	Obj.	Gap	Time	Infeasible Sol.	Infeasible Inst.	Obj.	Gap	Time
Medium	LKH3	-	0.00%	10.85	*	2.3h	-	0.00%	16.36	*	10.2h
	OR-Tools	-	100.00%	-	-	10.9s	-	100.00%	-	-	56.9s
	Random-L	99.96%	97.28%	21.02	138.56%	38s	100.00%	100.00%	-	-	2.0m
	Random-C	96.89%	47.39%	24.71	145.85%	37s	100.00%	99.98%	50.48	319.97%	2.0m
	PIP	1.75%	0.17%	11.23	3.59%	8s	2.50%	0.16%	17.68	8.10%	21s
	PIP-D	2.29%	0.22%	11.26	3.96%	8s	1.83%	0.23%	17.80	8.84%	23s
	LMask	**0.03%**	**0.01%**	**11.14**	**2.75%**	6s	**0.20%**	**0.05%**	**17.04**	**4.24%**	15s
Hard	LKH3	-	0.00%	13.25	*	2.6h	0.00%	0.00%	20.76	*	15.8h
	OR-Tools	-	100.00%	-	-	10.6s	-	100.00%	-	-	56.8s
	Random-L	100.00%	99.96%	22.2	132.40%	37s	100.00%	100.00%	-	-	2.0m
	Random-C	99.90%	94.05%	25.55	135.68%	37s	100.00%	100.00%	-	-	2.0m
	PIP	4.83%	2.39%	13.63	3.42%	8s	29.34%	21.65%	22.35	12.87%	20s
	PIP-D	4.16%	0.82%	13.79	4.28%	8s	13.51%	8.43%	22.90	12.53%	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

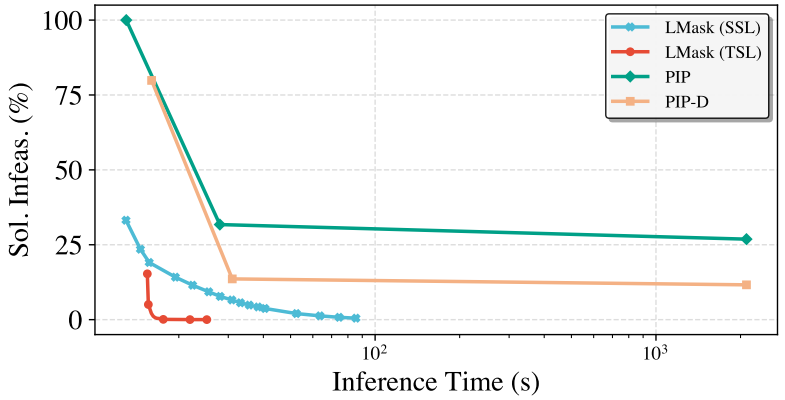
- Dumas et al. presented the well-known TSPTW benchmark dataset in 1995, which involves instances with various problem sizes and time window widths.
- We further evaluate all neural solvers on the TSPTW benchmark dataset.

Nodes	$n = 20$		$n = 40$		$n = 60$		$n = 80$	
Method	**Ins. Infeas.**	**Gap**	**Ins. Infeas.**	**Gap**	**Ins. Infeas.**	**Gap**	**Ins. Infeas.**	**Gap**
PIP	5%	5.2%	45%	4.6%	20%	11.5%	22%	8.7%
PIP-D	5%	5.2%	25%	6.3%	40%	13.1%	67%	12.0%
LMask	**0%**	**2.8%**	**0%**	**1.7%**	**0%**	**2.62%**	**0%**	**3.6%**

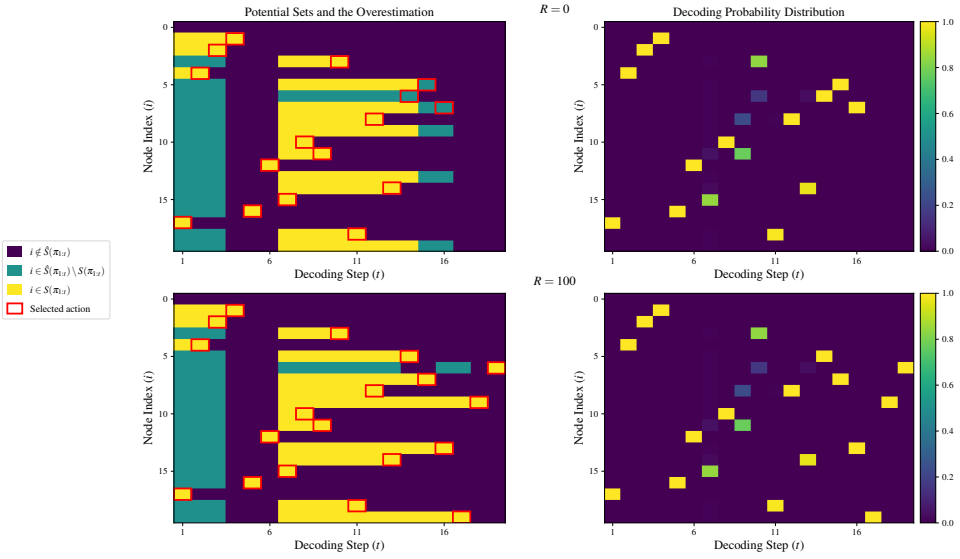
- Among all problem sizes, LMask achieves zero infeasibility rates and significantly smaller optimal gaps compared to PIP and PIP-D.

Lookahead vs. backtracking

- Relying solely on computationally expensive lookaheads creates a significant trade-off between mask accuracy and efficiency.
- Backtracking offers a flexible and powerful way to handle complex constraints because it enables the use of a lightweight overestimation strategies.



Visualization of the decoding process



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!

- T. Li*, H. Zou*, J. Wu, and Z. Wen, "LMask: Learn to solve constrained routing problems with lazy masking," ICLR, 2026.
- https://iclr.cc/virtual/2026/poster/10006412