Latent Guided Sampling for Combinatorial Optimization

Sobihan Surendran[1,2], Adeline Fermanian[2], and Sylvain Le Corff[1]

[1]Sorbonne Université and Université Paris Cité, CNRS, Laboratoire de Probabilités, Statistique et Modélisation, F-75005 Paris, France
[2]LOPF, Califrais' Machine Learning Lab, Paris, France

SCAN ME

Goal

☞ Introduce a novel latent space model for **Neural Combinatorial Optimization** that learns a structured latent representation.

☞ Propose an efficient inference method based on **Markov chain Monte Carlo** in the latent space and **Stochastic Approximation** for test-time adaptation toward lower-cost solutions, with theoretical guarantees.

☞ Illustrate empirically on routing problems that our method achieves **competitive performance** among learning-based methods.

Introduction: Combinatorial Optimization

Given an instance $x = \{\mathsf{x}_i\}_{i=1}^n$, we aim to find a solution y^* that minimizes the associated cost function C:

$$y^* \in \operatorname*{argmin}_{y \in \mathsf{Y}} C(y, x) \ ,$$

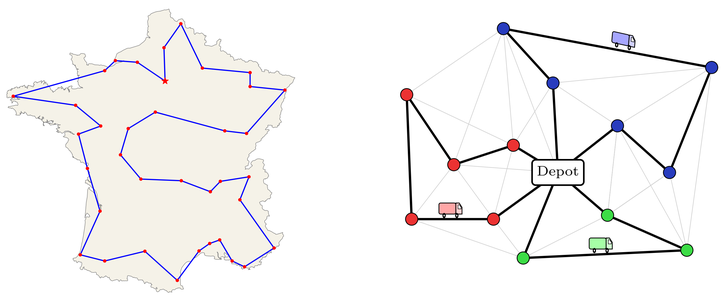
where Y denotes the set of all feasible solutions for the given problem.

Traveling Salesman Problem

- Find the shortest tour visiting each node exactly once.
- Input: $\mathsf{x}_i = c_i$, where c_i are the 2D coordinates of node i.

Vehicle Routing Problem

- Minimize the tour length while satisfying capacity constraints.
- Input: $\mathsf{x}_i = (c_i, d_i)$, where d_i is the demand of node i.



Prior Work

☞ **Pointer Networks** [1]: sequence-to-sequence models that construct solutions sequentially by selecting the next node to visit.

☞ **Attention Models** [2]: transformer-based policies trained with reinforcement learning to select nodes sequentially.

☞ **Latent generative approaches**: Conditional Variational Autoencoders (CVAE) [3] and Generative Adversarial Network (GAN)-inspired methods trained without a discriminator [4].

Limitations. ➤ Cost of obtaining labeled solutions for supervised training.
➤ Lack of structured latent space.
➤ Simple sampling or task-specific inference tricks.

Our Model Architecture

☞ Our model learns a **continuous latent representation** by modeling the **target distribution** to generate a solution y given an instance x:

$$p_{\text{target}}(y \mid x) = \int p_\phi(z \mid x) p_\theta(y \mid x, z) \, \mathrm{d}z \ .$$

☞ The probability of the decoder generating a solution y is factorized as:

$$p_\theta(y \mid x, z) = \prod_{t=1}^{T} p_\theta(\mathsf{y}_t \mid \mathsf{y}_{0:t-1}, x, z) \ ,$$

where $\mathsf{y}_t \in \{0, \dots, n\}$ is the selected node at step t.

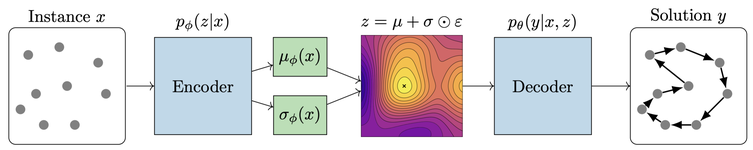


Figure: Our Latent Model Architecture with the Gaussian Reparameterization Trick.

Training Objective: Minimize the weighted cost while encouraging diversity among the generated solutions to improve inference efficiency:

$$\mathcal{L}(\theta, \phi; x) = \sum_{k=1}^{K} \mathbb{E}_{z^k \sim p_\phi(\cdot|x)} \Bigg[\underbrace{\mathbb{E}_{y^k \sim p_\theta(\cdot|x,z^k)} \left[w^k C(y^k, x) \right]}_{\text{weighted cost}} - \beta \underbrace{\mathcal{H}(p_\theta(\cdot \mid x, z^k))}_{\text{entropic regularization}} \Bigg] \ .$$

Inference

Our goal in inference is to sample from the distribution:

$$\pi_\theta(z, y \mid x) \propto p_\phi(z \mid x) \, p_\theta(y \mid x, z) \underbrace{e^{-\lambda C(y,x)}}_{\text{biasing towards lower-cost solutions}} \ .$$

Algorithm Latent Guided Sampling

Input: number of iterations M, number of particles K, and temperature schedule λ.
Initialize: Sample initial particles $z_0^k \sim p_\phi(\cdot \mid x)$ for all $k = 1, \dots, K$.
for $m = 0, 1, \dots, M-1$ **do**
  Propagate new particles: $\tilde{z}_{m+1}^k \sim q(\cdot \mid z_m^{1:K})$ for all $k = 1, \dots, K$.
  Generate candidate solutions: $y_{m+1}^k \sim p_{\theta_m}(\cdot \mid x, \tilde{z}_{m+1}^k)$ for all $k = 1, \dots, K$.
  Accept $z_{m+1}^k = \tilde{z}_{m+1}^k$ with probability: **Sampling**

$$\alpha_{m+1}^k = \min\left(1, \frac{p_\phi(\tilde{z}_{m+1}^k \mid x)}{p_\phi(z_m^k \mid x)} \exp\left[-\lambda\big(C(y_{m+1}^k, x) - C(y_m^k, x)\big)\right]\right) .$$

  Compute the gradient estimate $H_{\theta_m}\big(x, \{(z_{m+1}^k, y_{m+1}^k)\}_{k=1}^K\big)$. **Learning**
  Update parameters: $\theta_{m+1} = \theta_m - \gamma_{m+1} H_{\theta_m}\big(x, \{(z_{m+1}^k, y_{m+1}^k)\}_{k=1}^K\big)$.
end for

⇒ Use interacting MCMC to guide latent particles toward high-probability regions while encouraging exploration of the latent space.

⇒ Adapt the decoder at test time via stochastic gradient descent to favor lower-cost solutions.

Convergence Analysis

☞ **Fixed** θ. Geometric ergodicity: $\exists\, \rho_1, \rho_2 \in (0,1)$, $\kappa_1, \kappa_2 > 0$ s.t. $\forall\, m \in \mathbb{N}$:

$$\|\mu P_\theta^m - \pi_\theta\|_{\text{TV}} \le \kappa_1 \rho_1^m \ , \qquad \|\mu P_\theta^m - \pi_\theta\|_{L_2} \le \kappa_2 \rho_2^m \|\mu - \pi_\theta\|_{L_2} \ .$$

☞ **Adaptive** θ. ⇒ time-inhomogeneous Markov chain.

Convergence under parameter drift

$$\|\theta_m - \theta_\infty\|_{L_2} \xrightarrow[m\to\infty]{} 0 \ \Rightarrow \ \mathbb{E}[\|\mu P_{\theta_1} \cdots P_{\theta_m} - \pi_{\theta_\infty}\|_{\text{TV}}] \xrightarrow[m\to\infty]{} 0 \ .$$

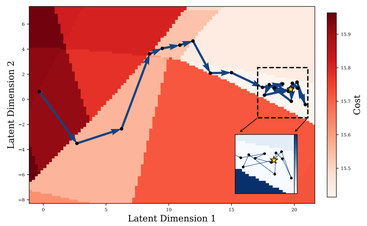
Experiments on CVRP

Table: Experimental results on CVRP: "Gap" denotes the average gap to baseline LKH3, and "Time" indicates the total runtime for solving 1000 instances.

	Training distribution		Generalization			
	n = 100		n = 125		n = 150	
Method	Gap	Time	Gap	Time	Gap	Time
LKH3	0.00%	17H	0.00%	19H	0.00%	20H
OR Tools	9.936%	38M	3.063%	64M	10.349%	73M
HGS	-0.505%	5H33	-0.723%	6H57	-0.879%	8H20
POMO (greedy)	1.287%	<1M	2.314%	<1M	3.444%	<1M
POMO (sampling)	0.431%	40M	0.869%	1H	1.659%	1H30
CVAE-Opt	1.364%	32H	2.080%	36H	3.240%	46H
EAS	0.148%	40M	0.234%	1H	0.515%	1H30
COMPASS	0.135%	40M	0.263%	1H	0.718%	1H30
ELG	1.261%	40M	1.308%	1H	1.540%	1H30
CNF	0.328%	40M	1.040%	1H	4.047%	1H30
LGS-Net (ours)	**-0.102%**	40M	**-0.022%**	1H	**0.343%**	1H30

Inference Comparison and Latent Space Illustration

Method	Gap	Time
Sampling	0.721%	40M
DE	0.135%	40M
CMA-ES	0.271%	40M
EAS	0.933%	40M
Adam	0.592%	40M
SGLD	0.431%	40M
Single MCMC	0.701%	40M
Parallel MCMC (ours)	0.109%	40M
Interacting MCMC (ours)	-0.032%	40M
LGS (ours)	**-0.102%**	40M



☞ **Perspective:** Scale LGS-Net to larger and more constrained routing problems, and develop hybrid approaches combining learning-based methods with classical heuristics.

References

[1] Irwan Bello et al. "Neural combinatorial optimization with reinforcement learning". In: *International Conference on Learning Representations, Workshop Track*. 2017.

[2] Wouter Kool, Herke Van Hoof, and Max Welling. "Attention, learn to solve routing problems!" In: *International Conference on Learning Representations*. 2019.

[3] André Hottung, Bhanu Bhandari, and Kevin Tierney. "Learning a latent search space for routing problems using variational autoencoders". In: *International Conference on Learning Representations*. 2021.

[4] Felix Chalumeau et al. "Combinatorial optimization with policy adaptation using latent space search". In: *Advances in Neural Information Processing Systems*. Vol. 36. 2023, pp. 7947–7959.