MONASH University

CP 2026
Lisbon · FLoC 2026

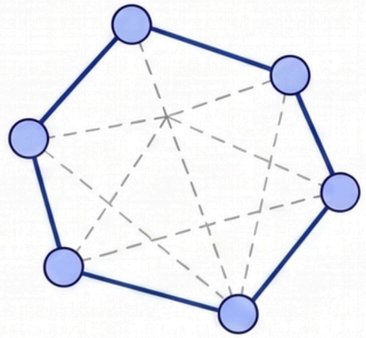
Constraint-Aware Self-Supervised Learning for Edge Selection

Xinda Zheng*, Frits de Nijs, Edward Lam

Background

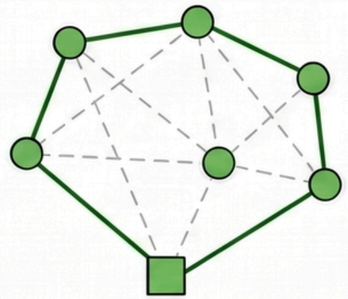
TSP
(Traveling Salesman Problem)

Minimum-cost
Hamiltonian cycle



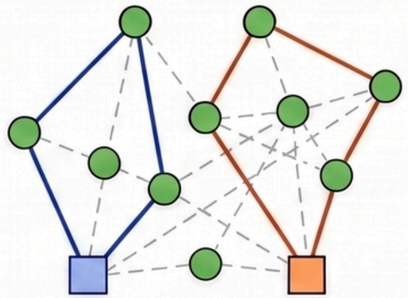
OP
(Orienteering Problem)

Visit **rewarding nodes** under **budget**



TOP
(Team Orienteering Problem)

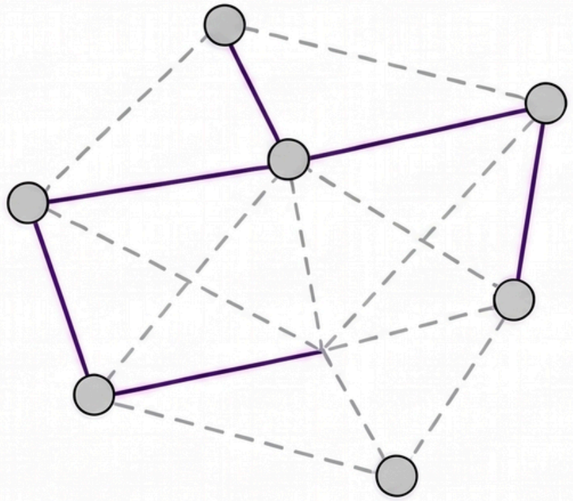
Multiple vehicles visit rewarding nodes under **budgets**



••• shortest path, VRP, scheduling, ...

Unified formulation

Unified Edge Selection View
(General Formulation)



Given a graph with

node features + **edge features** + **global features**
e.g., budgets, #vehicles, depot, time limit, ...

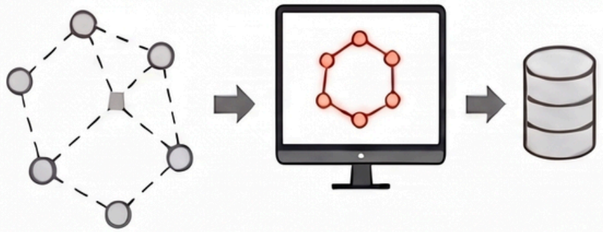
learn to select edges
to optimize an objective
under **problem-specific constraints**

Different objectives and constraints, but the same underlying decision: **selecting edges**.

Background

1 Training Labels

Optimal or high-quality labels

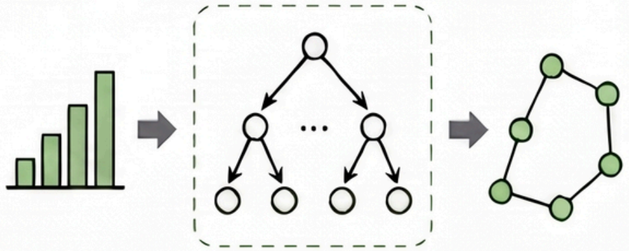


- Requires pre-solving instances for training
- Label becomes cost as instance size or dataset scale grows

Supervised Cost

2 Heavy Post-Processing

Sampling, search, or refinement

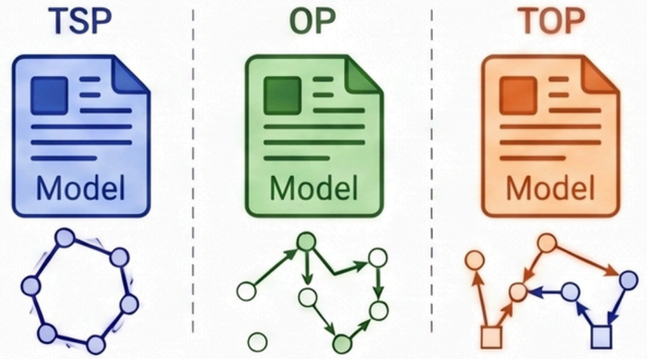


- Additional inference-time computation
- Solution quality depends on search effort

Heavily relies on post-processing

3 Narrow Problem-Specific Designs

Separate pipelines across problems



- Overly tailored to specific problems or constraint structures
- Hard to transfer knowledge or reuse across problem classes

Limited Reuse

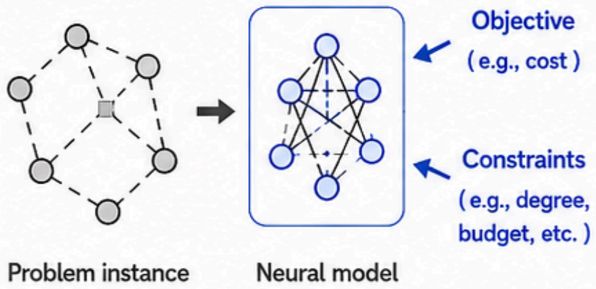
Can we learn edge selections **without solution labels** or **expensive search**, using a **reusable framework**?

Our Approach

1 Constraint-Aware Self-Supervision

Learn directly from objectives and constraints

No solution labels

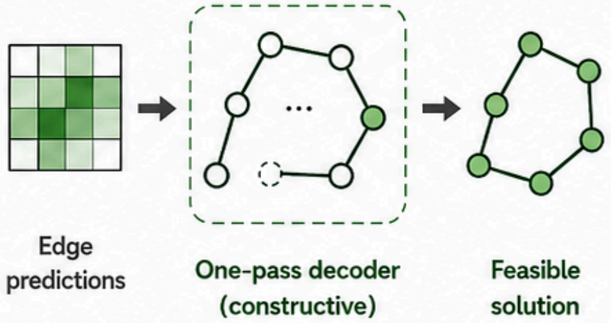


- No optimal/heuristic labels
- Train on many unlabeled instances with objective and constraints

2 One-Pass Feasible Construction

Convert edge predictions into feasible solutions

No iterative search or refinement

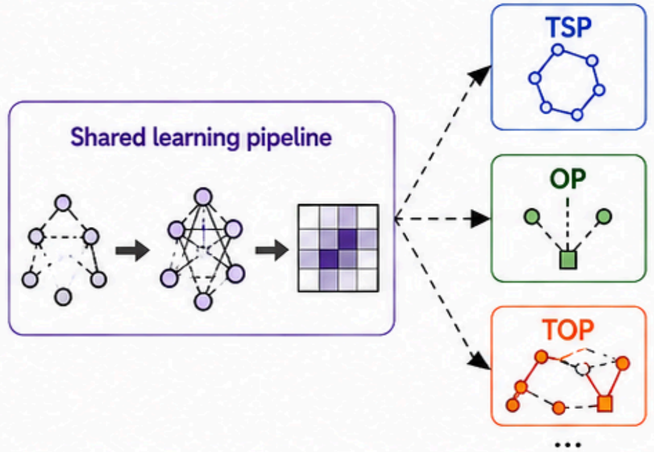


- Lightweight one-pass decoding
- Avoid heavy search, sampling, repair, or refinement

3 Reusable Edge-Selection Framework

Shared learning pipeline across problems

Task-specific objectives, constraints, and decoders



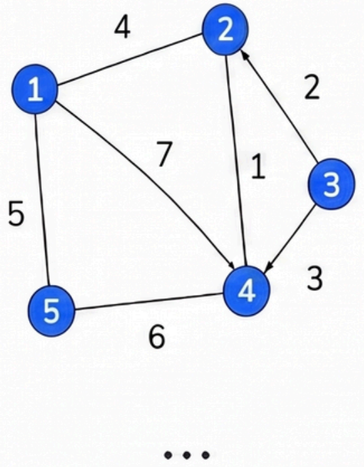
- Share representations and modules
- Adapt to different problems via objectives, constraints, and decoders

Learn constraint-aware solution representations directly—without labels or heavy downstream post-processing.

Framework Overview

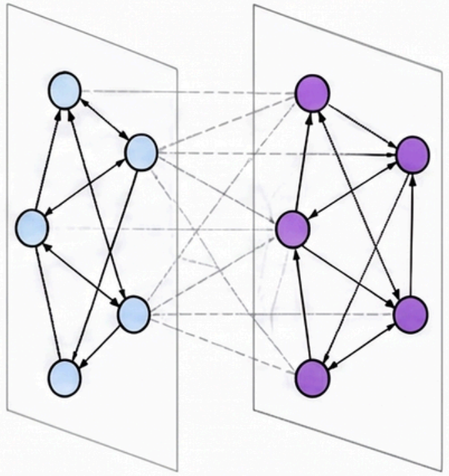
1. Problem Instance

Graph + features



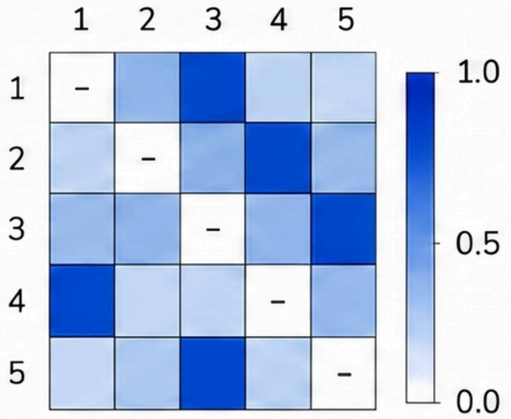
2. Neural Model

Cost-Attention Convolution
+ edge scoring



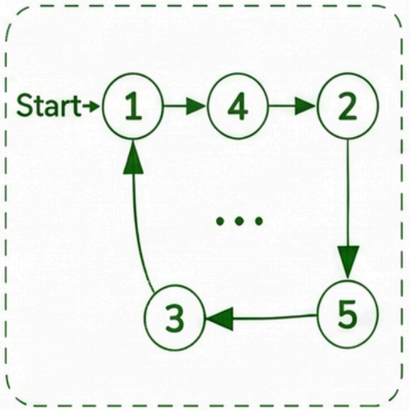
3. Transition Probability P

Learned solution
representation



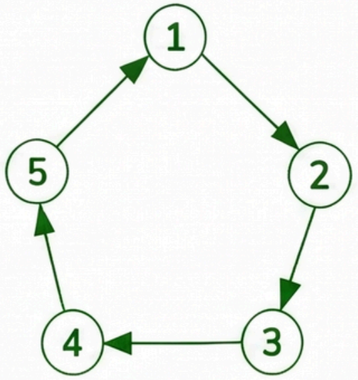
4. One-Pass Decoder

Constructive,
constraint-aware



5. Feasible Solution $\hat{x}$

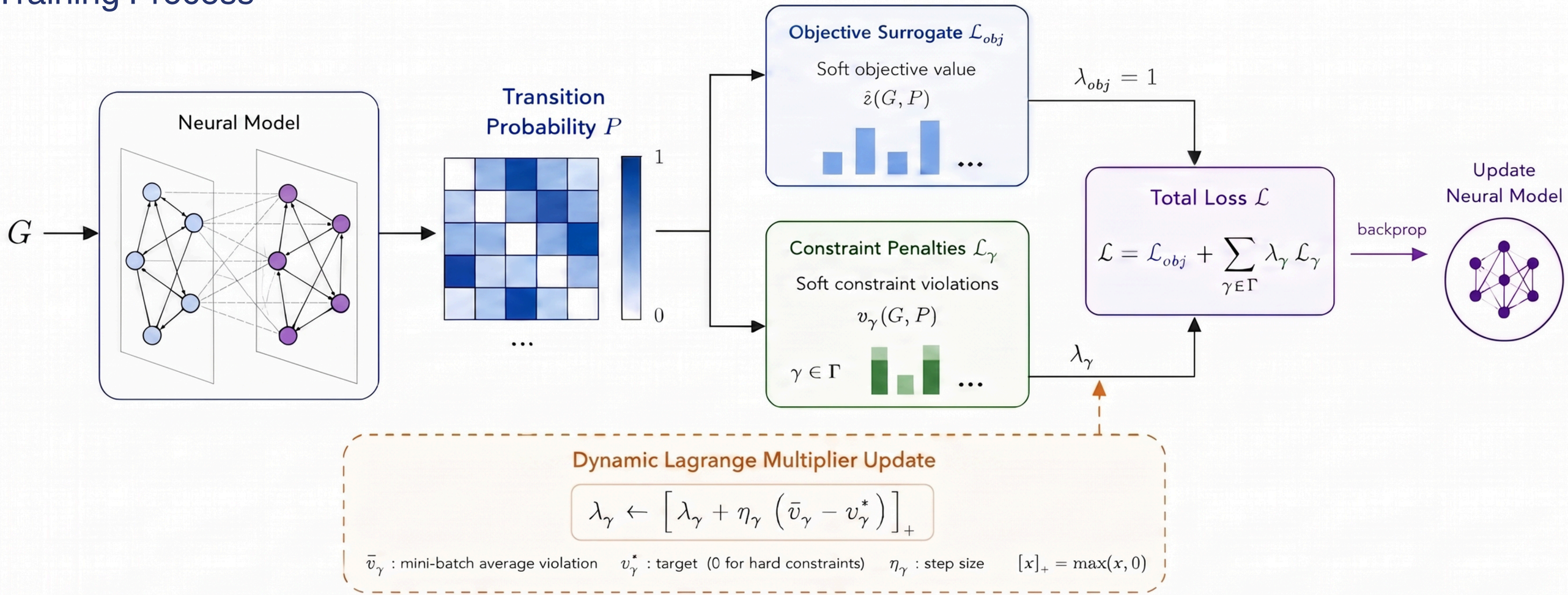
Satisfies problem
constraints



Self-supervised training signal

Objective surrogate + Constraint penalties + Dynamic Lagrange multipliers

Training Process



Dynamic Lagrange Multiplier Update

$$\lambda_{\gamma} \leftarrow \left[\lambda_{\gamma} + \eta_{\gamma} \left(\bar{v}_{\gamma} - v_{\gamma}^{*} \right) \right]_{+}$$

$\bar{v}_{\gamma}$: mini-batch average violation v_{γ}^{*} : target (0 for hard constraints) η_{γ} : step size $[x]_{+} = \max(x, 0)$

1) Objective Surrogate $\mathcal{L}_{obj}$

Differentiable approximation of problem objective.

e.g., expected tour cost, expected total reward, etc.

2) Constraint Penalties $\mathcal{L}_{\gamma}$

Penalize soft violations of problem constraints.

e.g., degree, subtour elimination, budget, capacity, etc.

3) Total Loss $\mathcal{L}$

Balances objective quality and constraint satisfaction.

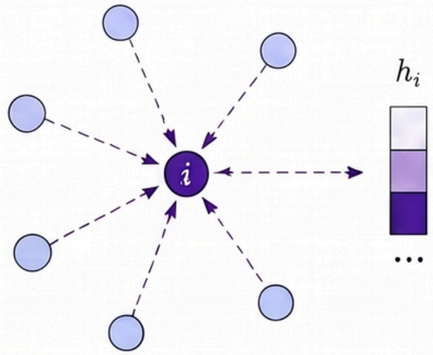
Driven by learned multipliers (no solution labels).

Model

1 GNN on Dense Graphs: Motivation

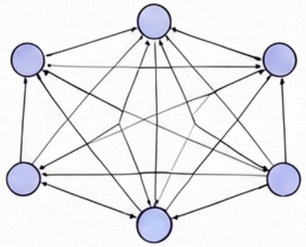
What does a GNN do?

Each node aggregates information from its neighbors.



Why is this challenging on dense graphs?

Many edge-selection problems are represented as dense graphs.



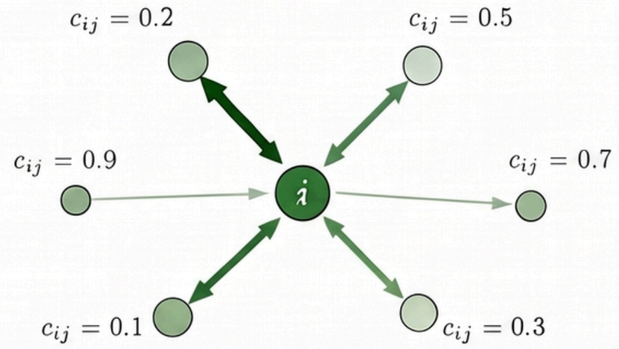
All node pairs interact
(many pairwise connections)

Edge-wise message passing
introduces substantial computational overhead

2 Cost-Attention Convolution (CAC)

Use the cost matrix to determine how strongly information flows between nodes.

A. Intuition: Cost determines attention



Low cost → high weight

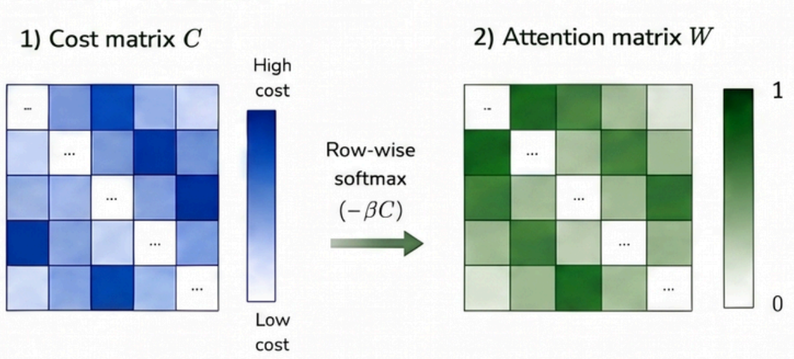
High cost → low weight

Attention weight

$$W_{ij} \propto e^{-\beta C_{ij}}$$

Lower cost ⇒ larger attention weight

B. Matrix formulation



C. Message aggregation for node i

$$m_i = \sum_{j=1}^{n} W_{ij}\, r_j$$

Node i aggregates neighbor features r_j weighted by cost-aware attention W_{ij}.

Cost-aware representation of node i

Leverages available cost structure

Reduces irregular edge-wise overhead

Lightweight and easy to train

Case Study

Shared architecture and training principle, with task-specific losses and decoders.

What is shared?

Same neural architecture (CAC + bilinear scoring)

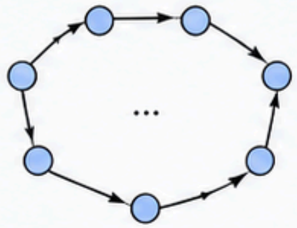
Same way of producing transition probabilities over edges

Same self-supervised training with objective surrogate + constraint penalties

Same single-pass decoding principle to ensure feasibility

Task-Specific Instantiation

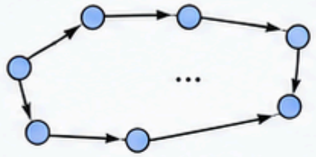
TSP
Traveling Salesman Problem



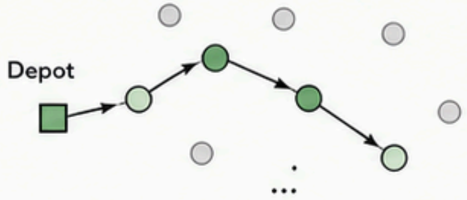
Objective Signal
Minimize expected travel cost

Constraint Signal
Visit each node once
Subtour elimination

Decoder
Degree-constrained tour construction (2-regular cycle)



OP
Orienteering Problem

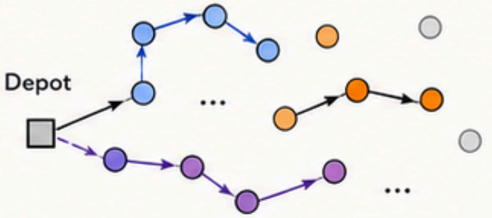


Objective Signal
Maximize collected reward

Constraint Signal
Budget constraint
Depot usage
No revisits

Decoder
Budget-feasible route construction from depot

TOP
Team Orienteering Problem



Objective Signal
Maximize shared reward

Constraint Signal
Per-vehicle budget
Cross-vehicle assignment
No revisits

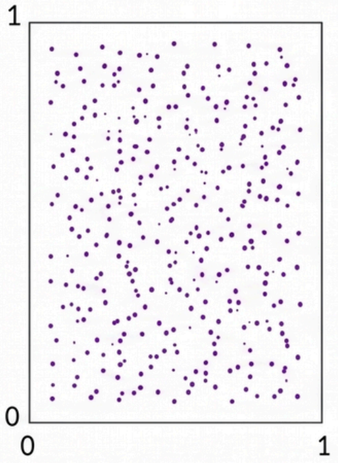
Decoder
Sequential multi-route construction with assignment

Experimental Setup

Dataset

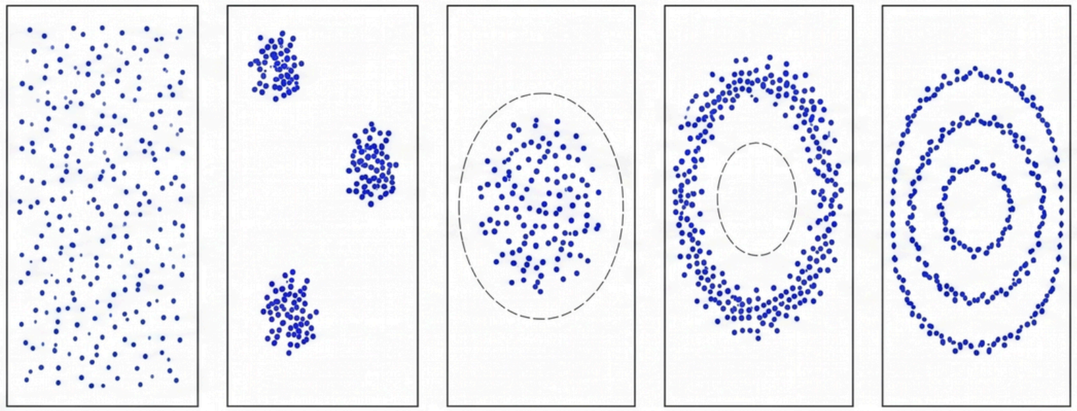
Training Dataset

Uniform



Test Datasets

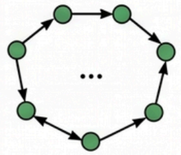
Uniform | Clusters | Ball | Donut | Waves



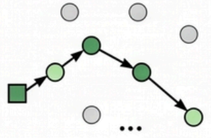
- Our model is trained on 10,000 unlabeled uniform instances.
- Models are evaluated on both in-distribution data and four unseen geometric layouts to test generalization.

Tasks

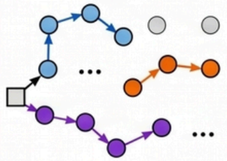
TSP (100 / 200)



OP (100)



TOP (100) K=5



Comparisons

All tasks

- Greedy (construction heuristic)
- Gurobi (optimization reference)

Optimal for TSP/OP: 1% gap for TOP

TSP-specific baselines

Learning-based:

- UTSP (full pipeline + search/refinement)
- $UTSP_1$ (one-pass version)
- AGFN (full pipeline + search/refinement)
- $AGFN_1$ (one-pass version)

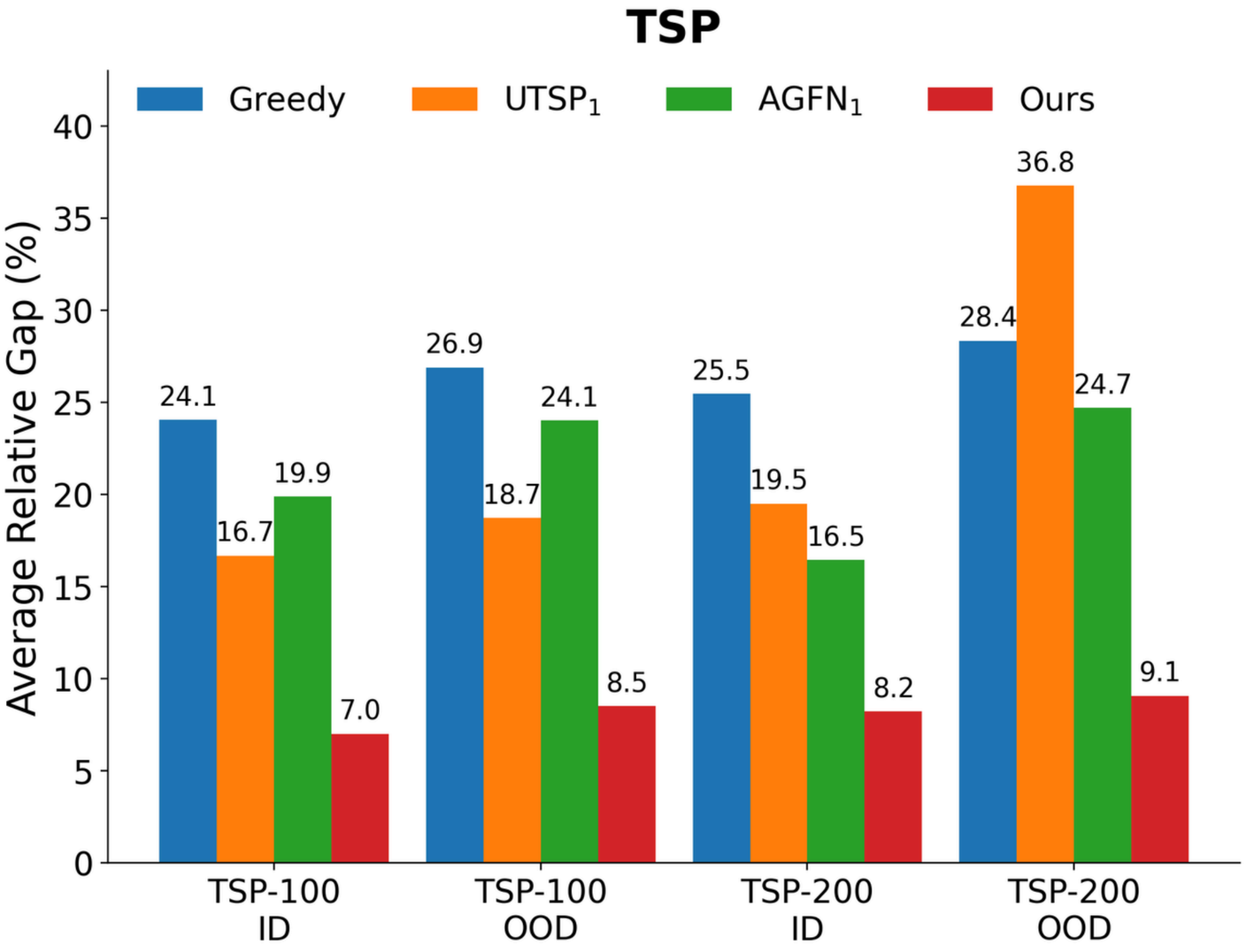
Specialized heuristic:

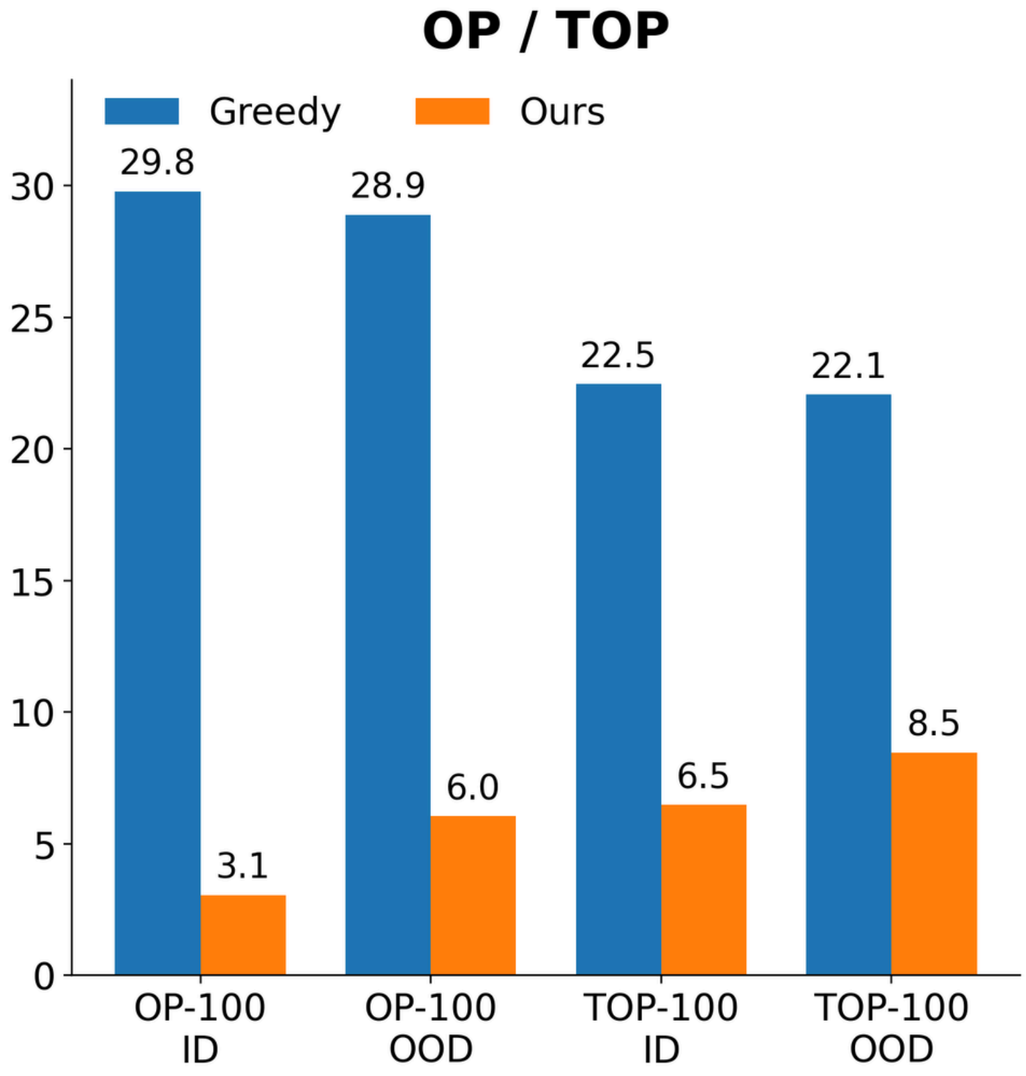
- LKH3 (specialized heuristic)

Evaluation Metrics

- Average Relative Gap (%) (lower is better)
- Worst-Case Relative Gap (%) (lower is better)
- Inference + Decoding/Search Time (Total Time)

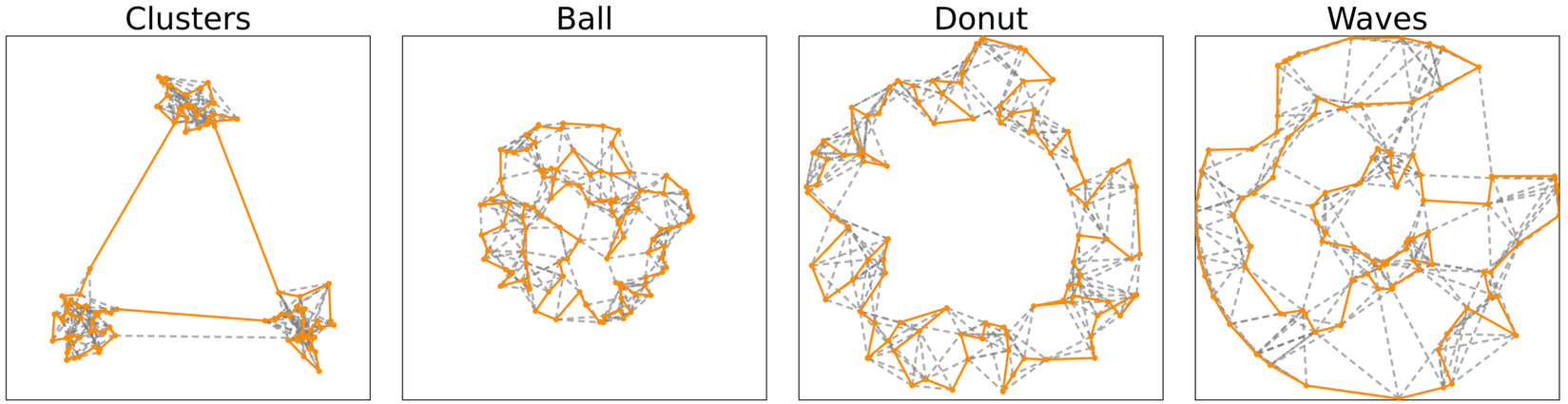
Results: Performance





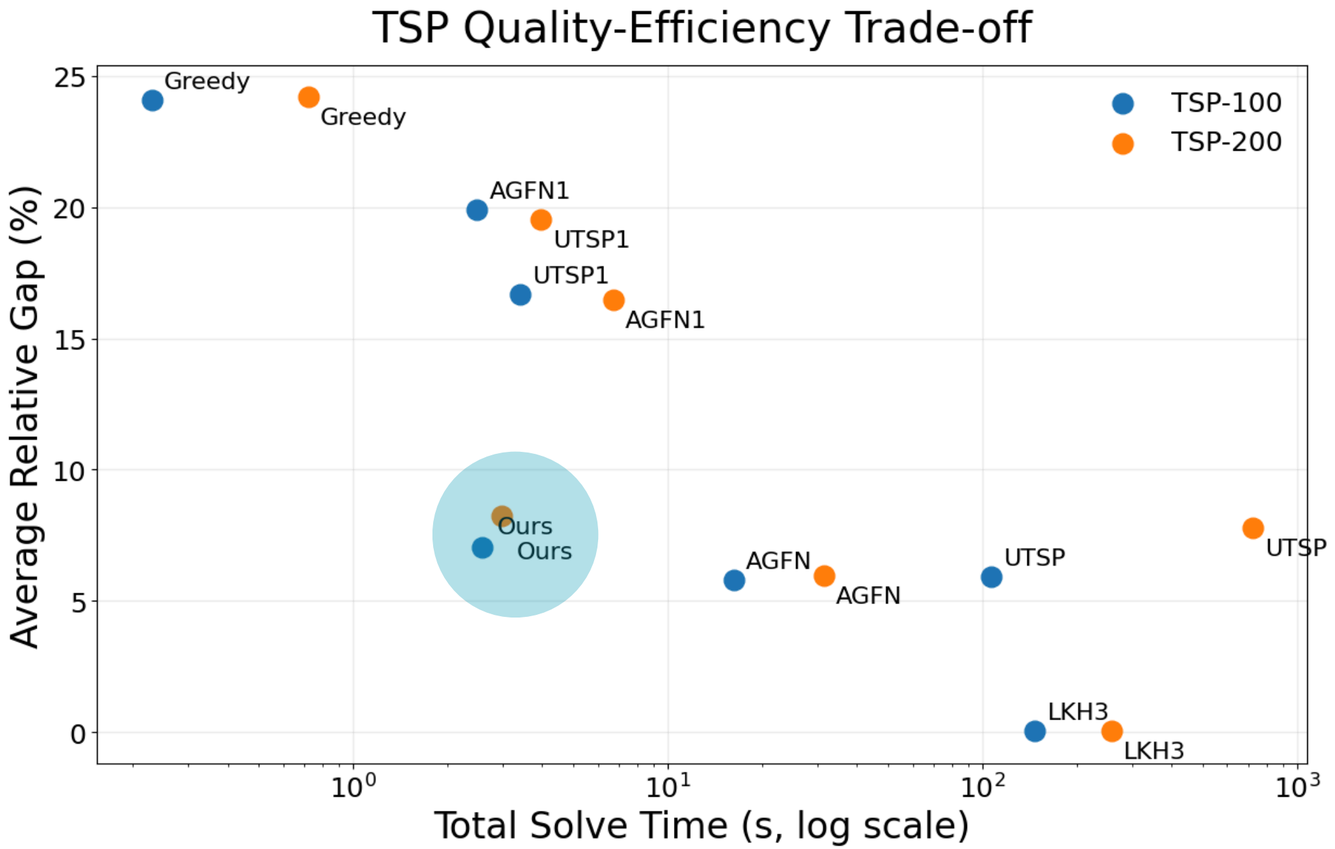
- *Best one-pass solution quality across three routing problems*
- *Generalizes well beyond the training distribution*

Results: TSP



- *Predicted high-probability edges align with optimal tour structure across unseen layouts*

- Orange: Optimal tour
- Gray dashed: Top-3 predicted edges



- *Best solution quality among lightweight one-pass methods*
- *Much faster than search-heavy learned pipelines and specialized solvers*

Conclusion & Future Work

A reusable self-supervised framework that learns objective- and constraint-aware solution representations directly from unlabeled instances.

- Bridge the training-inference gap between continuous representations and discrete solutions
- Learn richer constraint structures with less task-specific engineering
- Bridge toward end-to-end prediction and optimization

Thank you