Neural Sequence Generation with Constraints via Beam Search with Cuts: A Case Study on VRP

POUYA SHATI, ELDAN COHEN, SHEILA MCILRAITH

UNIVERSITY OF TORONTO

VECTOR INSTITUTE

SoCS'24 + Doctoral Consortium

Summary

Motivation

Neural sequence generation can solve combinatorial optimization problems
- Lacks support for hard constraints
- Lacks guarantee when using Beam Search

Vehicle Routing Problems (VRP) are important combinatorial tasks
- Involve global constraints that require meticulous reasoning
- Existing neural methods do not support global constraints

Contributions

Beam search with cuts
- Combines any pre-trained neural model with CSP requirements

Two requirements applicable in multiple settings
- Bin Packing in encoded IP
- Regular Language encoded in SAT
- Solve 3 VRP variants with hard constraints

Results

Satisfies requirements with negligible cost to quality

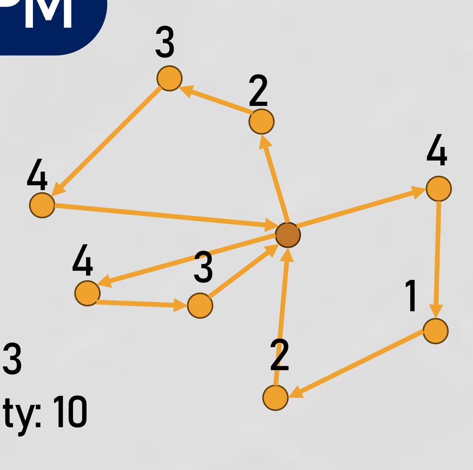
Scales exponentially better when problem size increases

Vehicle Routing Problems

Nodes: $N = \{n_i \mid n_i \in \mathbb{R} \times \mathbb{R}\}$

Objective: minimize total distance

CVRPM



Tours: 3

Capacity: 10

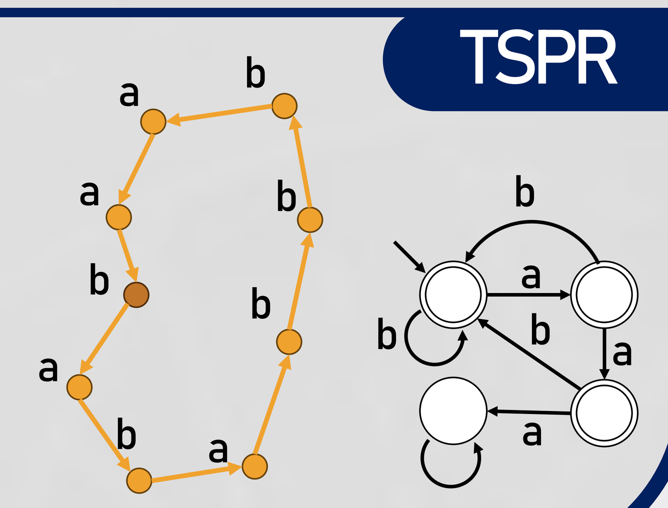
Constrained Vehicle Routing Problem with Maximum Tours:

Find a series of $m \in \mathbb{N}$ tours from depot n_d partitioning the nodes, s.t. the sum of demands $D: N \to \mathbb{N}$ in each route is less than the capacity $c \in \mathbb{N}$

TSPR

Travelling Salesman Problem with regular specification

Find one complete tour x s.t. $\sigma(x) \in \mathcal{A}$, where Σ_A is an alphabet, $\sigma: N \to \Sigma_A$ an alphabet mapping, and $\mathcal{A}$ a Deterministic finite automata (DFA)



[1] Kool, W.; van Hoof, H.; and Welling, M. 2018. Attention, Learn to Solve Routing Problems! In ICLR.

[2] Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning.

[3] Uchoa, E.; Pecin, D.; Pessoa, A.; Poggi, M.; Vidal, T.; and Subramanian, A. 2017. New benchmark instances for the capacitated vehicle routing problem. EJOR.

Presentation: Friday 7th, Session 3, 12pm

Framework

Beam Search (BS)

Generate a sequence x from tokens Σ

Neural Model: next token prediction function $\mathbf{p}: \Sigma^* \to \mathcal{P}(\Sigma)$

Optimize sequence score $\theta(x) = \Pi_i\, p(x_1, x_2, \dots, x_i)[x_{i+1}]$

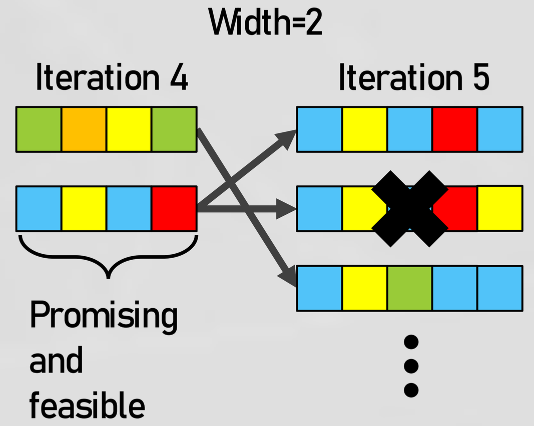
Sets of partial solutions of size i: S_i

Beam width (w): number of partial solutions

$$S_i = argmax_{1:w}(\{\theta(x.a) \mid x \in S_{i-1}, a \in \Sigma\})$$

start → Initialize with empty solution → w promising partial solutions → All possible expansions → Candidates for next step → Top w scoring candidates → w promising partial solutions → Solutions are complete → Final solutions

Beam Search with Cuts (BSC)



Explicitly checks feasibility of partial solutions

Impedes infeasible partial solutions from expanding further

$$S_i = argmax_{1:w}\left(\{\theta(x.a) \mid x \in S_{i-1}, a \in \Sigma, \exists x' : x.a.x' \in R \wedge [x.a.x' \text{ is complete}]\}\right)$$

Similar procedure as BS With added checks

start → Initialize with empty solution → w promising partial feasible solutions → All possible expansions → Candidates for next step → Top w scoring feasible candidates → w promising partial feasible solutions → Solutions are complete and feasible → Final solutions

Solve CSP → ✓ Add / ✗ Cut

Requirements

Bin Packing

Partition items I with weights $W: I \to \mathbb{N}$ into m bins of capacity c

Nodes, Demands, Tours, Capacity

Encoded in Boolean Satisfiability

Neural model solving CVRP + Bin Packing Requirement = Solving CVRPM

Regular Language

Given alphabet Σ_A, alphabet, mapping $\sigma: N \to \Sigma_A$ and DFA $\mathcal{A}$, find x s.t. $\sigma(x) \in \mathcal{A}$

Encoded in Integer Programming

Neural model solving TSP + Regular Language Requirement = Solving TSPR

Experiments

Setup

Kool et al. [1]
- Uses attention layers and is trained using REINFORCE [2]
- Solves CVRP and TSP
- Used in beam search with cuts as a pre-trained neural model
- Used in beam search with large width as baseline

Solvers:
- IP: Gurobi
- SAT: Gluecard 4

Timeout limit:
- 10 seconds per CSP call

Datasets:
- Uchoa et al. [3]
- Synthetic, following [1]

Sequence Generation with Requirements

BSC Satisfies all requirements
- Even when BS fails

With negligible cost to quality
- Up to 24% improvement

With smaller width and less runtime

27 instances from Uchoa et al. → BS Width =8k → 18 satisfied ✓ / 9 unsatisfied → BSC Width =4 → satisfied ✓

Tightening Requirements

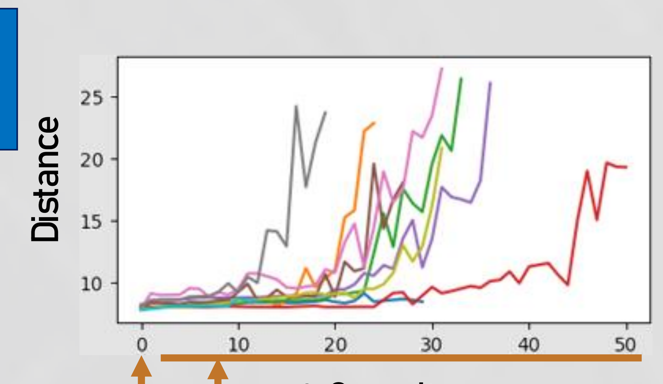
Tightest requirement satisfied by BS is detected, as a starting point

BSC can tighten requirements with negligible cost to quality

Quality and tightness trade-off until infeasibility

BS Width =8k

BSC Width =4

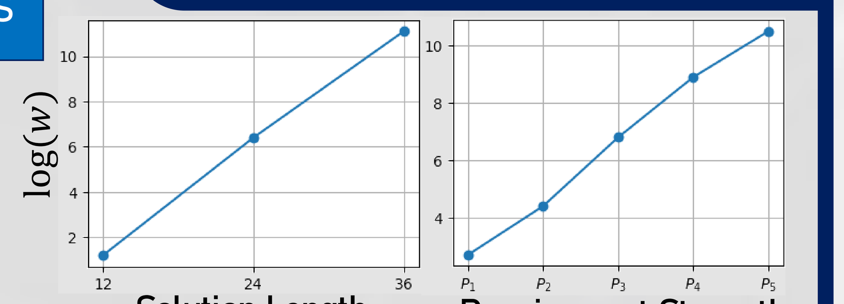


Scaling Problem Size

Reported lowest width needed for requirement satisfaction

BS scales exponentially
- As solution length or requirement strength is increased

BS



BSC Width =4

BSC's performance remains stable
- Solves all instances in 1.73-1.95 seconds

Future Work

Add cuts due to equivalency checks between partial solutions

- Cache and query feasibility - Cut dominated solutions - Increase solution diversity

Application to neural models for other problems

Implementation of new requirements

Integration with Beam-stack search to enable backtracks