Reduced VNS - Sampling, Algorithm and Shaking

Purpose:
Fast exploration without local search refinement.

Algorithm

- Initialize: best = initial solution, $k = 1$
- While $k \leq k_{\text{max}}$:
  - candidate = shake(best, k)
  - If cost(candidate) $<$ cost(best):
    - best = candidate
    - $k = 1$
  - Else, $k = k + 1$
- Return best

Shaking Function (Random m-exchange)

- Select m customers and randomly reinsert elsewhere.
- $S' = \text{Shake}(S, m) = \text{RandomInsert}(\text{Select}(S, m))$.
- Accept if $C(S') < C(S)$.

Why Random m-exchange Works

- Cheap to compute and always feasible.
- Explores new areas of the solution space.
- Degree of shake controlled by parameter m.

Basic VNS - Neighborhood Expansion, Algorithm, and 2-opt Local Search

Purpose: Avoid local minima by structured neighborhood enlargement.

Algorithm

- Initialize: best = initial solution, $k = 1$
- While $k \leq k_{\text{max}}$:
  - shaken = shake(best, k)
  - local_opt = local_search(shaken)
  - If cost(local_opt) < cost(best):
    - best = local_opt
    - $k = 1$
  - Else, $k = k + 1$
- Return best

2-opt Local Search

- Applied intra-route to break and reconnect edges to eliminate crossings.
- For route $r = (v_0, \ldots, v_i, v_{i+1}, \ldots, v_j, v_{j+1}, \ldots, v_n)$, reversal gives:

$$r' = (v_0, \ldots, v_i, v_j, \ldots, v_{i+1}, v_{j+1}, \ldots, v_n)$$

- Cost improvement:

$$\Delta C = d(v_i, v_j) + d(v_{i+1}, v_{j+1}) - d(v_i, v_{i+1}) - d(v_j, v_{j+1})$$

- Accept move if $\Delta C < 0$.
- Lightweight and effective local optimizer.

Skewed VNS - Cost and Algorithm

Problem: Reduced VNS and Basic VNS may stagnate in local minima.

Skewed Cost Function:

$$C_{\text{skew}}(S') = C(S') + \alpha \cdot d(S', S_{\text{best}})$$

where:

- $C(S')$ is the cost of candidate solution S'.
- $d(S', S_{\text{best}})$ measures the distance between S' and the best solution found.
- α controls the tradeoff between cost and diversity.

Algorithm:

- Initialize: best = initial solution, $k = 1$
- While $k \leq k_{\text{max}}$:
  - shaken = shake(best, k)
  - local_opt = local_search(shaken)
  - skewed_cost = cost(local_opt) + $\alpha\times$ solution_distance(local_opt, best)
  - If skewed_cost ¡ cost(best), then
    - best = local_opt
    - $k = 1$
  - Else, $k = k + 1$
- Return best

Skewed VNS Effectiveness and Solution Distance Function

Why Skewed VNS is Effective:

- Penalizes solutions too similar to current best.
- Promotes exploration of distant basins of attraction.

Why symmetric difference?

- Captures node assignment and position mismatch.
- Reflects structural route changes.
- Efficient to compute vs edit distance or embeddings.

Solution Distance Function:

$$d(S_1, S_2) = \left| \bigcup_{r \in S_1} \mathrm{PosPairs}(r) \triangle \bigcup_{r' \in S_2} \mathrm{PosPairs}(r') \right|$$

where

$$\mathrm{PosPairs}(r) = \{(i, \text{node}) \mid \text{node at position } i \text{ in route } r\}.$$

Solution Distance Function - Python Pseudocode:

- Define `pos_pairs(routes)` as the set of (position, node) pairs for all routes.
- Compute symmetric difference:

$$d = |\text{pos\_pairs}(routes_1) \triangle \text{pos\_pairs}(routes_2)|$$

VNS Comparative Summary and Insights

Comparative Analysis of VNS Variants

Feature	RVNS	BVNS	SVNS
Shake	Random m-exchange	Random m-exchange	Random m-exchange
Local Search	No	2-opt	2-opt
Neighborhood Control	Simple	Systematic	Systematic + Penalized
Exploration	Shake	Shake + Local	Shake + Local + Diversity
Acceptance Criterion	$C(S') < C(S)$	$C(S') < C(S)$	$C_{\text{skew}} < C_{\text{best}}$
Distance Function	No	No	Yes
Computational Cost	Low	Medium	Med-High
Effectiveness	Fast	Better	Best

Takeaways & Usage

- **RVNS:** Lightweight, stochastic. Use for real-time or quick heuristics.
- **BVNS:** Adds local improvement via 2-opt. Use for standard optimization routines.
- **SVNS:** Promotes exploration through diversity. Use for complex search spaces.

Future Directions

- Adaptive α tuning based on performance.
- Learnable and dynamic solution representations.
- Hybridization with deep reinforcement learning agents.
- Online adjustment of neighborhood structures.