Solution Strategy

Break problem into 2 parts:

- Find initial feasible solution
- Optimize

Initial Solution

- No concern for optimality → so basically, a binpacking problem
- Easy to model using CP

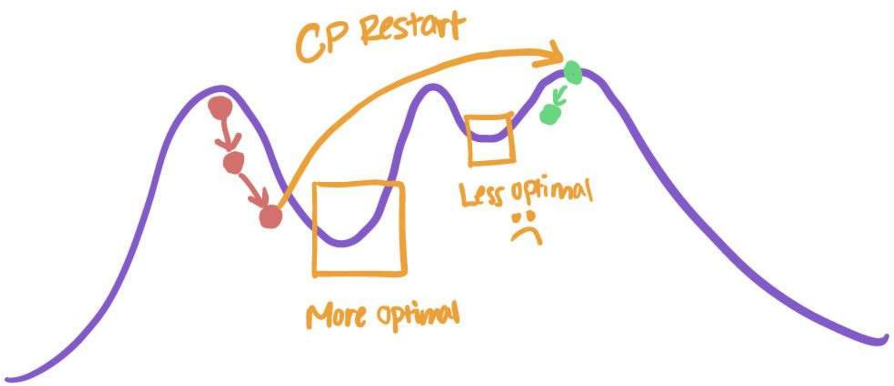
Optimization

- Local search

Initial Local Search Strategy (Not Ideal)

- Define multiple moving strategies
- Get **large neighborhood** from each moving strategy and pick the most optimal solution that is feasible
- Exploration
  - **If no new incumbent found within time limit, resolve with CP and continue LS from there**
- Exploitation
  - Large neighborhood due to considering many candidate solutions enables exploitation

Problem: Not enough exploitation using work done so far!

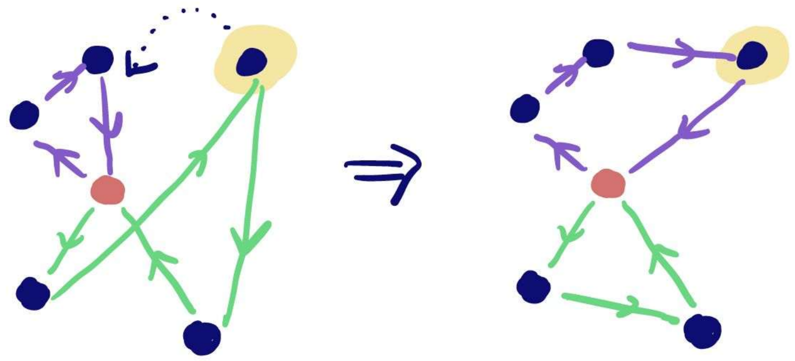


Improved Local Search Strategy

- Define multiple moving strategies
- Get **a single neighbor** from each moving strategy and pick the most optimal solution that is feasible
- **Tolerance**
  - Allow moves to less optimal solutions within tolerance
  - If no new incumbent found within time limit, **move back to incumbent solution (to continue optimizing from there) and decrease tolerance**
- Exploration
  - Early on, high tolerance + smaller neighborhood enables exploration of search space
- Exploitation
  - Later on, low tolerance forces movement towards (local) minima

Neighborhood Definition

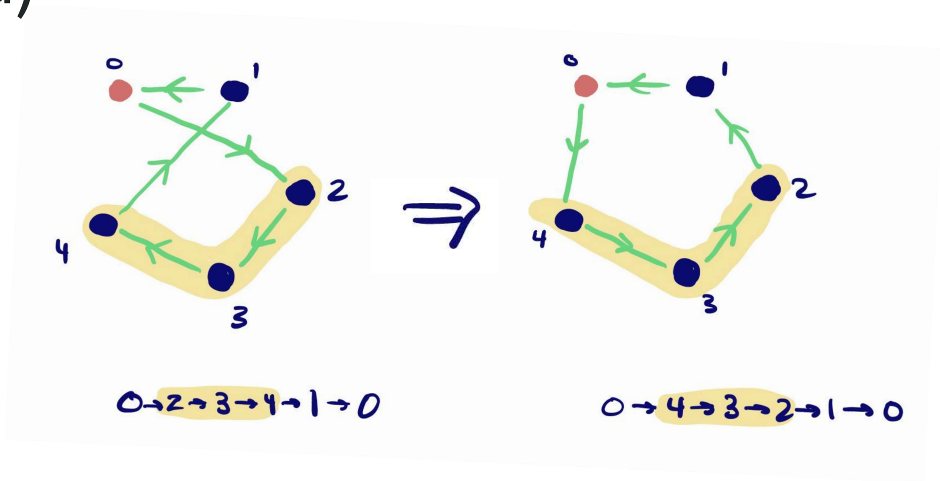
Strategy 1:
Move a random customer from current route to a random position in a different (randomly picked) route



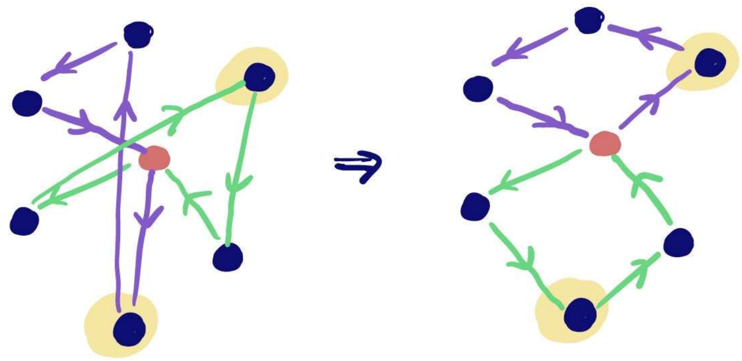
Strategy 2:
Move random customer to a random location in a randomly picked route (could be within the same route)

Neighborhood Definition (cont'd)

Strategy 3 (Two-Opt):
Reverse the order of a random subpath in a route



Strategy 4:
Randomly swaps two customers in different routes



Implementation Techniques

Speed up candidate solution evaluation by using diff (vs previous solution) to compute feasibility, total distance

- Moves in solution space only impact 1-2 routes; therefore, feasibility and distance change computations can be limited to an analysis of those changes

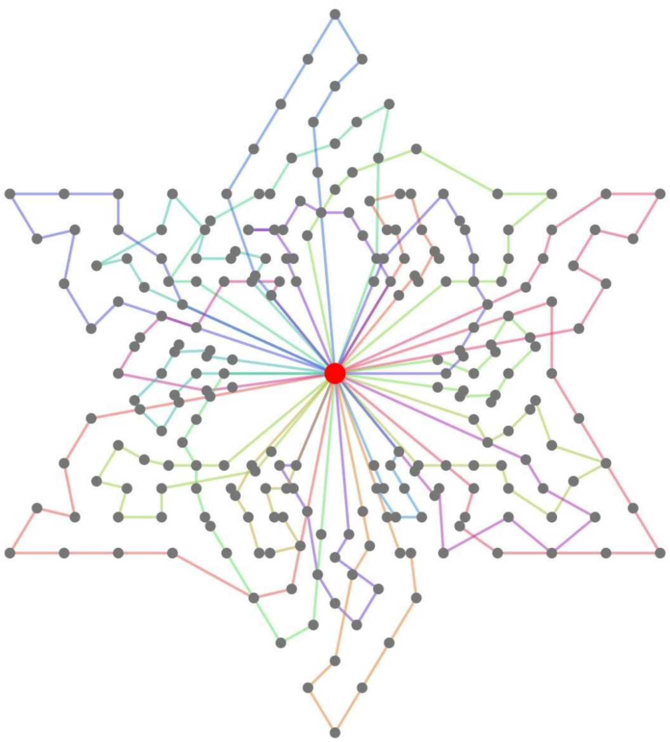
Other Things We Tried (that didn't work)

Multithreading

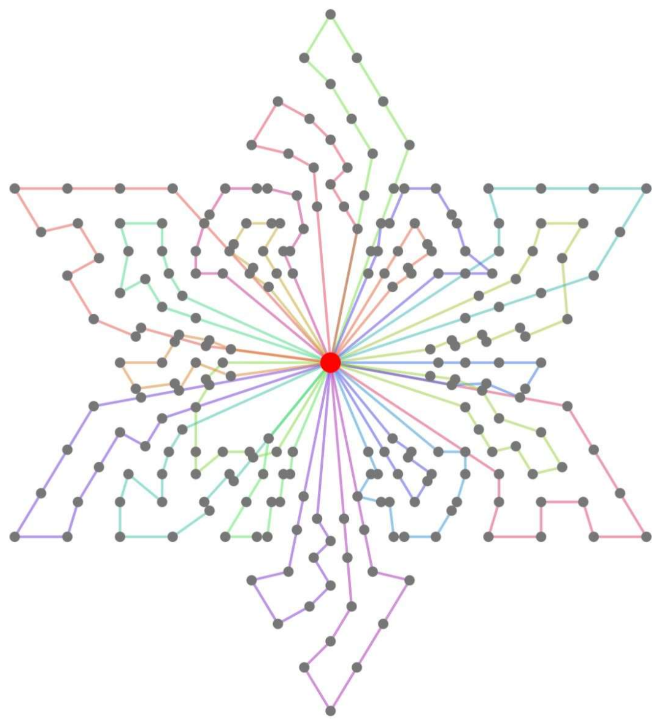
- Helped with large neighborhoods
- Smaller neighborhoods along with optimized feasibility and distance checking for candidate solutions rendered this useless

Experimental Observations (241_22_1.vrp)

OR Tools (after 30 seconds):



Our Solution (after 295 seconds):



Experimental Observations (Verification)

- Outperforms Google OR-Tools on every instance
- Verified feasibility and total distance for solutions separately

	OR-Tools (after 295 seconds)	Our Solution (after 295 seconds)	Percentage Difference
16_5_1.vrp	337	334.96	-0.61%
51_5_1.vrp	533.5	524.61	-1.67%
151_15_1.vrp	3137.5	3103.27	-1.09%
262_25_1.vrp	5850	5679.32	-2.92%
386_47_1.vrp	26934	25940.84	-3.69%

Note: lower values are better since we are trying to minimize the total distance traveled.