Spring 2026

Large-scale Transportation Optimization

Cathy Wu

1.041/1.200 Transportation: Foundations and Methods

Readings

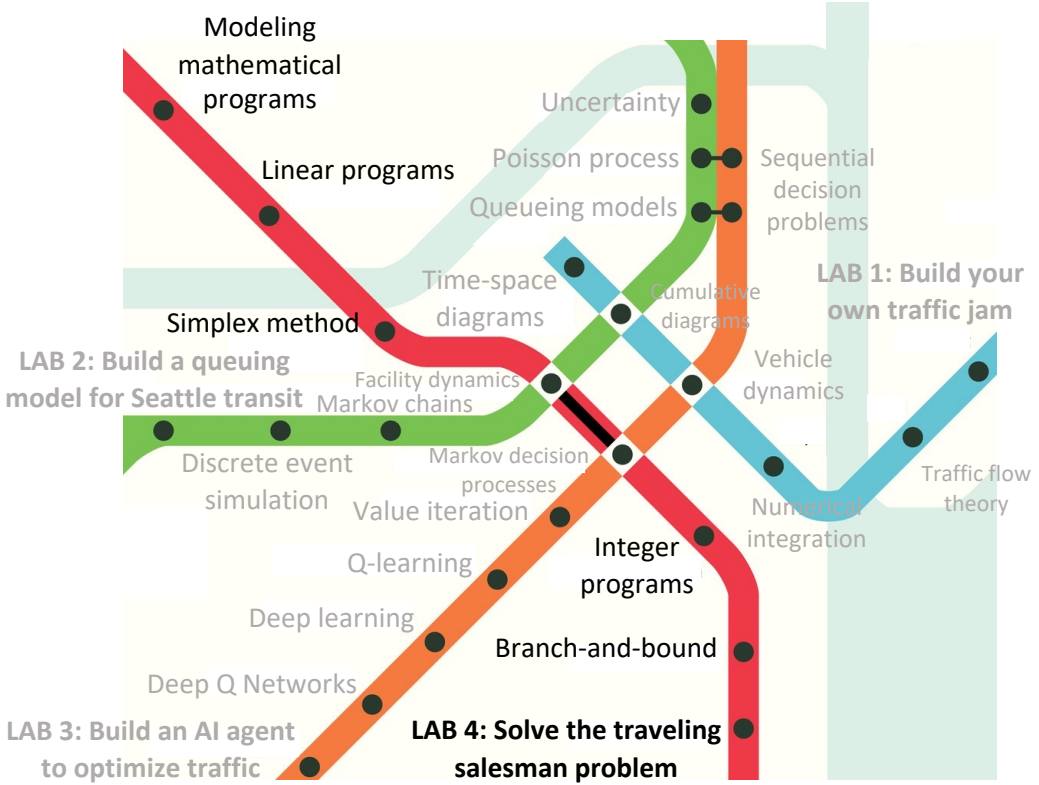
1. G. Laporte, “Fifty Years of Vehicle Routing,” *Transportation Science*, vol. 43, no. 4, pp. 408–416, Nov. 2009, doi: 10.1287/trsc.1090.0301.
2. (Optional) G. Laporte, S. Ropke, and T. Vidal, “Chapter 4: Heuristics for the vehicle routing problem,” in *Vehicle routing: Problems, methods, and applications, second edition*, SIAM, 2014, pp. 87–116. [pdf]

Unit 4: Optimizing transportation resources

Unit 4

Optimizing

Single-stage



Where this lecture sits

Today: why exact formulations are useful, where they hit scalability limits, and how heuristics take over.

Already covered

- LP modeling and geometric intuition
- Reduced costs and LP solving
- IP modeling and branch-and-bound

Learning goal today

- Recognize when transportation models outgrow generic MILP
- Understand construction heuristics, local search, and metaheuristics
- Use CVRP as a running example

Outline

1. Transportation optimization beyond generic MILP
2. Running example: capacitated vehicle routing (CVRP)
3. Construction heuristics
4. Local search and neighborhoods
5. Metaheuristics
6. Advanced vehicle routing

Transportation optimization beyond generic MILP

Recall: Traveling salesman problem

Notation
- $x_{ij} \in \{0,1\}$, 1 if a vehicle travels directly from i to j.
- c_{ij}: travel cost from i to j

$$\text{Minimize} \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$$

subject to:

$$\sum_{i=1}^{n} x_{ij} = 1 \quad (j = 1, 2, \dots, n)$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad (i = 1, 2, \dots, m)$$

- Sub-tours are not prevented.

From exact models to scalable methods

Why MILP is attractive

- Expressive modeling language
- Exact solutions and optimality bounds
- Flexible constraints

Why scale hurts: **branching can be exponential in problem size**

- Binary decisions grow quickly with network size
- Valid routes and schedules are defined by global structure, not local arcs
- Real systems need high-quality solutions fast and repeatedly

Key message

- Transportation practitioners use a spectrum: exact polynomial algorithms → MILP → specialized exact methods → heuristics/metaheuristics.

Running example: capacitated vehicle routing (CVRP)

Running example: Vehicle Routing Problem (VRP)

- One of the most widely studied in Combinatorial Optimization:
  - ≈10,000 works published in 2025 alone (Google Scholar), mostly heuristics
  - Direct application in real systems that distribute goods and provide services
  - Last-mile delivery, waste collection, field service, ride-pooling, on-demand transit

~10,000

papers in 2025

$11T+

global logistics costs, 2023

65M+

US parcel deliveries / day

Dantzig, G. B., & Ramser, J. H. (1959). *The Truck Dispatching Problem*. Management Science, 6(1), 80–91
Stats: Statista/Research & Markets (2024); ShipMatrix US parcel report (2024).

Vehicle Routing Problem (VRP)

Reflecting the variety of real transportation systems, the VRP literature is spread across hundreds of variants.

Classical dimensions

- Vehicle capacities
- Time windows
- Heterogeneous fleets
- Multiple depots
- Pickup and delivery, backhauling
- Split delivery
- Arc routing (e.g., garbage collection)

Emerging variants

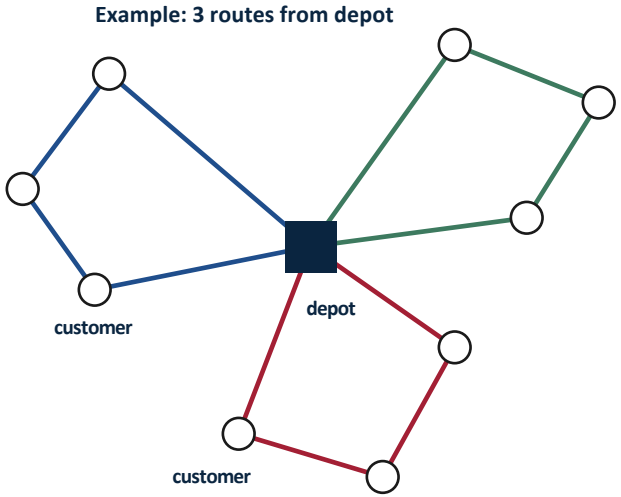
- Electric vehicle routing (charging decisions)
- Drone routing (3D, endurance limits)
- Warehouse routing (pickers, sortation)
- Stochastic and dynamic VRP (real-time requests)
- Learning-augmented routing (ML-guided heuristics)
- Multi-modal logistics (truck + drone, rail + truck)

Capacitated Vehicle Routing Problem (CVRP)

- **First** (Dantzig and Ramser [1959]) and **most basic multi-vehicle variant**:

- **Instance**: Complete graph $G = (V, E)$ with $V = \{0, \dots, n\}$; 0 is the depot, $V_c = \{1, \dots, n\}$ is the set of customers. Each edge $e = (i, j) \in V \times V$ costs c_{ij}. Each $i \in V_c$ demands q_i units. Homogeneous unlimited fleet of vehicles with capacity Q.

- **Solution**: Routes from the depot, respecting the capacities and visiting all customers once; minimizing the total cost.



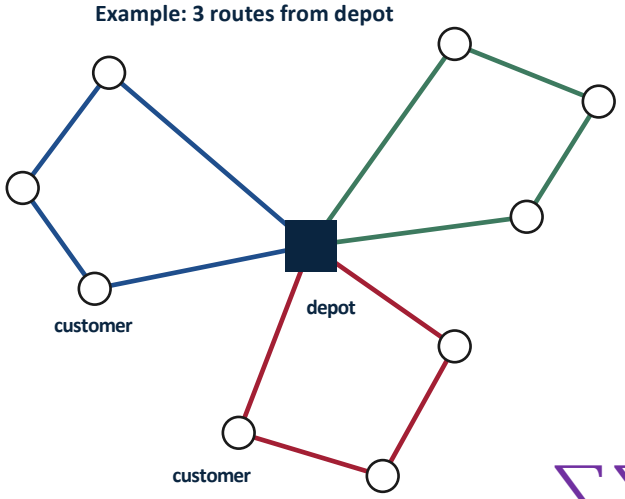
Capacitated Vehicle Routing Problem

We formulate the CVRP as a classic Integer Linear Program (ILP).

Objective: Minimize total travel distance

$$\text{Minimize} \sum_{i \in V} \sum_{j \in V, j \neq i} c_{ij} x_{ij}$$

subject to:



$$\sum_{i \in V, i \neq j} x_{ij} = 1 \quad \forall j \in V_c$$

$$\sum_{j \in V, j \neq i} x_{ij} = 1 \quad \forall i \in V_c$$

Degree Constraints: Visit each customer once

$$\sum_{j \in V_c} x_{0j} = \sum_{i \in V_c} x_{i0}$$

$$\sum_{j \in V_c} x_{0j} \geq \left\lceil \frac{\sum_{i \in V_c} q_i}{Q} \right\rceil$$

Depot Constraints: Flow balance at depot, and lower bound on vehicles

$$\sum_{i \in S} \sum_{j \notin S} x_{ij} \geq \left\lceil \frac{\sum_{i \in S} q_i}{Q} \right\rceil \quad \forall S \subseteq V_c, S \neq \emptyset$$

Capacity cuts: Eliminate subtours and enforce capacity (dynamically added → *branch-and-cut*)

Notation
- $V = \{0,1,\dots,n\}$ nodes, with depot 0
- $V_c = \{1,\dots,n\}$: customers
- $x_{ij} \in \{0,1\}$, 1 if a vehicle travels directly from i to j.
- c_{ij}: travel cost from i to j
- q_i: demand of customer i
- Q: vehicle capacity

Adapted from: Toth, P., & Vigo, D. (2002). *Models, relaxations and exact approaches for the capacitated vehicle routing problem*. Discrete Applied Mathematics.

VRP is NP-Hard

Difficult to solve via enumeration

Combinatorial Growth

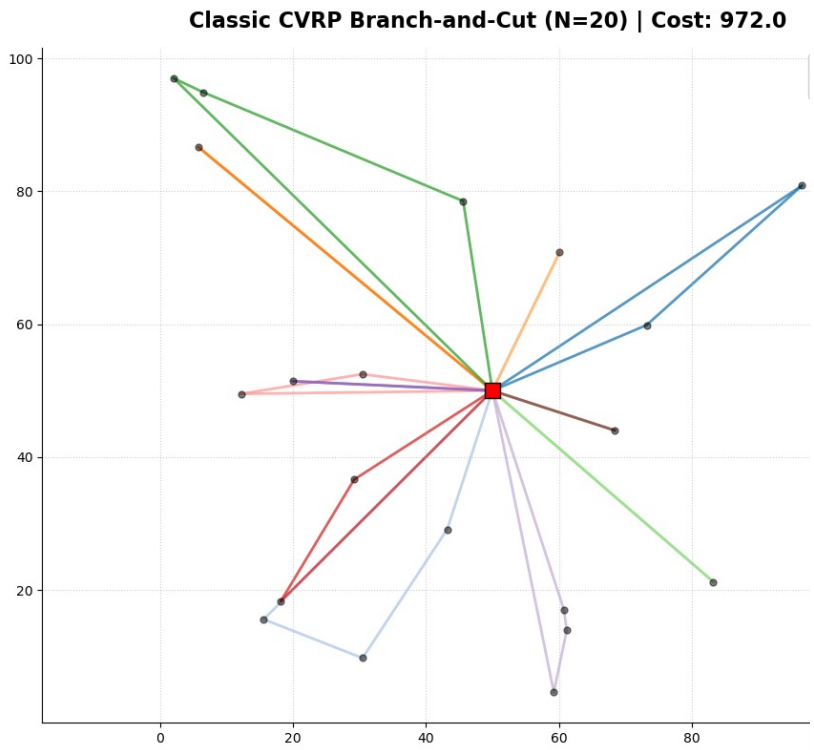
- 3 stops
- 5 stops
- 10 stops

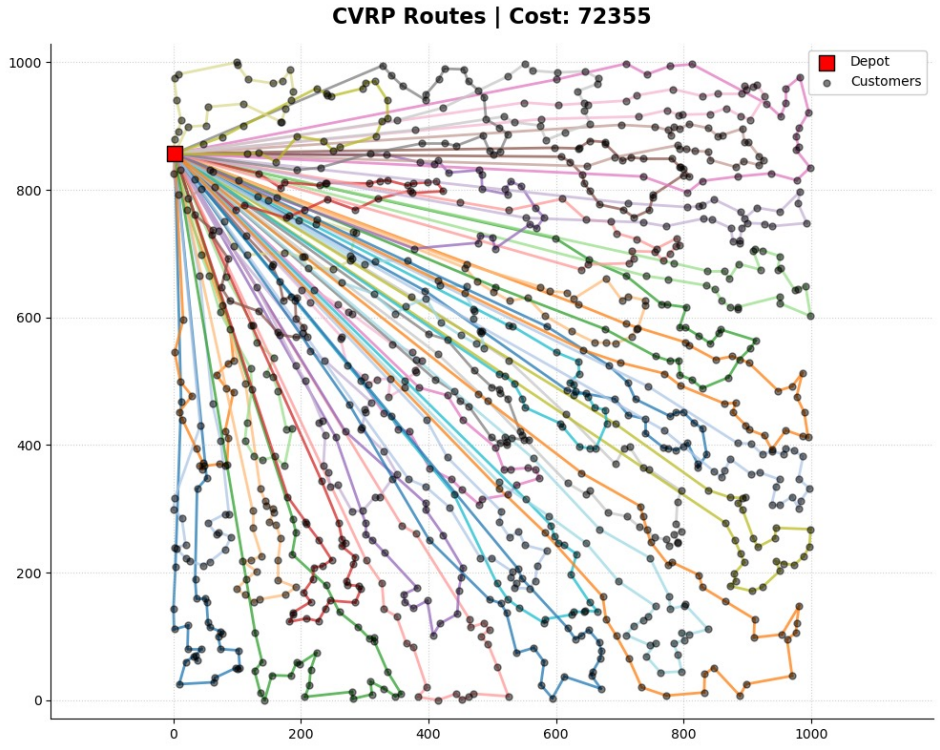
Customers total	Customers on the route	Ways to select customers for the route	Ways to select and sequence the route	Hours of work to evaluate one per second
10	3	120	720	0.20
20	3	1,140	6,840	1.9
30	3	4,060	24,360	6.8
40	3	9,880	59,280	16
50	3	19,600	117,600	33
60	3	34,220	205,320	57
70	3	54,740	328,440	91
80	3	82,160	492,960	137
90	3	117,480	704,880	196
100	3	161,700	970,200	270
				Days of work to evaluate one per second
10	5	252	30,240	0.35
20	5	15,504	1,860,480	22
30	5	142,506	17,100,720	198
40	5	658,008	78,960,960	914
50	5	2,118,760	254,251,200	2,943
60	5	5,461,512	655,381,440	7,585
70	5	12,103,014	1,452,361,680	16,810
80	5	24,040,016	2,884,801,920	33,389
90	5	43,949,268	5,273,912,160	61,041
100	5	75,287,520	9,034,502,400	104,566
total	on the route			**Years** of work to evaluate one per second
10	10	1	3,628,800	0.12
20	10	184,756	670,442,572,800	21,260
30	10	30,045,015	109,027,350,432,000	3,457,235
40	10	847,660,528	3,075,990,524,006,400	97,539,020
50	10	10,272,278,170	37,276,043,023,296,000	1,182,015,570
60	10	75,394,027,566	273,589,847,231,501,000	8,675,477,145
70	10	396,704,524,216	1,439,561,377,475,020,000	45,648,191,828
80	10	1,646,492,110,120	5,974,790,569,203,460,000	189,459,366,096
90	10	5,720,645,481,903	20,759,078,324,729,600,000	658,266,055,452
100	10	17,310,309,456,440	62,815,650,955,529,500,000	1,991,871,225,125

Adapted from Goentzel 2004

Capacitated Vehicle Routing Problem

- N=20 vs 1000





Construction heuristics

Construction heuristics

Goal: produce a feasible solution fast — a starting point that local search can then improve.

Nearest neighbor

Start at depot, repeatedly visit the closest unvisited customer that fits capacity. Close route when capacity exhausted.

Greedy, trivially simple, poor quality

Clarke–Wright savings

Start with one route per customer. Merge routes greedily by the savings achieved. Still the standard baseline.

Fast, classical, strong baseline

Sweep

Sort customers by polar angle from the depot; assign to routes in angular order until capacity is filled.

Geometric, good for clustered data

Insertion heuristics

Start with a seed tour; insert remaining customers one at a time at the cheapest feasible position.

Flexible, extends naturally to time windows

None guarantee optimality — but all produce a feasible solution in polynomial time.

Nearest-neighbor construction

Key idea

Grow one route greedily by repeatedly taking the closest feasible unvisited customer.

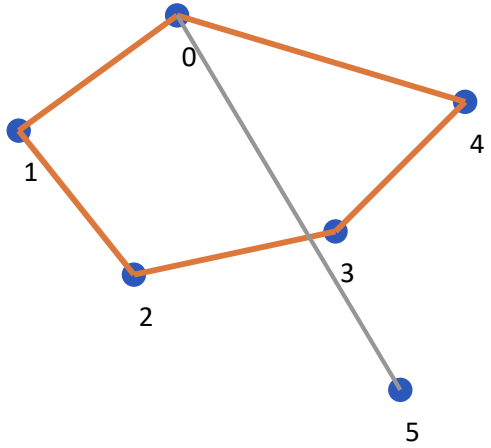
$$next(i) = \arg\min \{ c(i,j) : j \text{ unvisited and feasible} \}$$

Selection rule at current customer i

Algorithm

1. Start a route at the depot
2. Pick the nearest feasible customer
3. Repeat until no feasible customer fits
4. Return to depot and start the next route

Greedy view



Fast baseline

Myopic: nearby choices can strand far customers.

Sweep construction

Key idea

Exploit geometry: sort customers by polar angle around the depot, then pack them into routes in angular order.

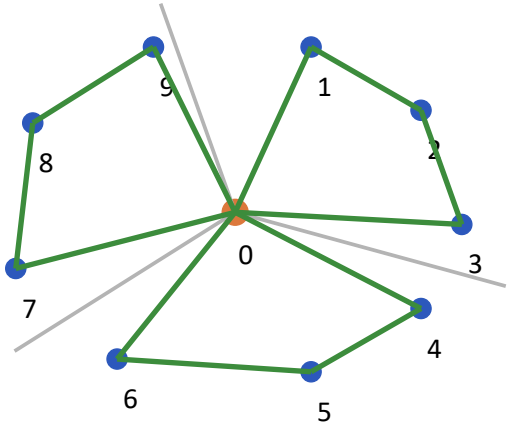
$$\boldsymbol{\theta_i = \operatorname{atan2}(y_i - y_0, x_i - x_0)}$$

Sort customers by angle θ_i

Algorithm

1. Compute each customer's angle around the depot
2. Sort customers by increasing angle
3. Sweep around the circle, adding customers until capacity is full
4. Start a new route and continue

Geometric view



Works well for clustered Euclidean instances.

Insertion heuristics

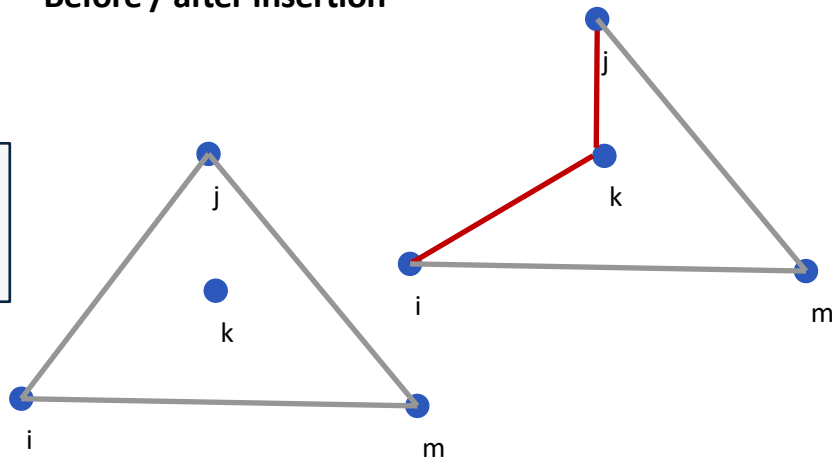
Key idea

Start from a seed route and insert customers one at a time where the marginal cost increase is smallest.

$$\Delta(i,k,j) = c(i,k) + c(k,j) - c(i,j)$$

Choose the feasible insertion with minimum Δ

Before / after insertion



Algorithm

1. Initialize a seed route (or one route per seed)
2. For each remaining customer, evaluate all feasible insertion positions
3. Insert the best customer-position pair
4. Repeat until all customers are placed

Flexible and easy to extend to time windows.

Clarke-Wright Savings (CW)

Key idea

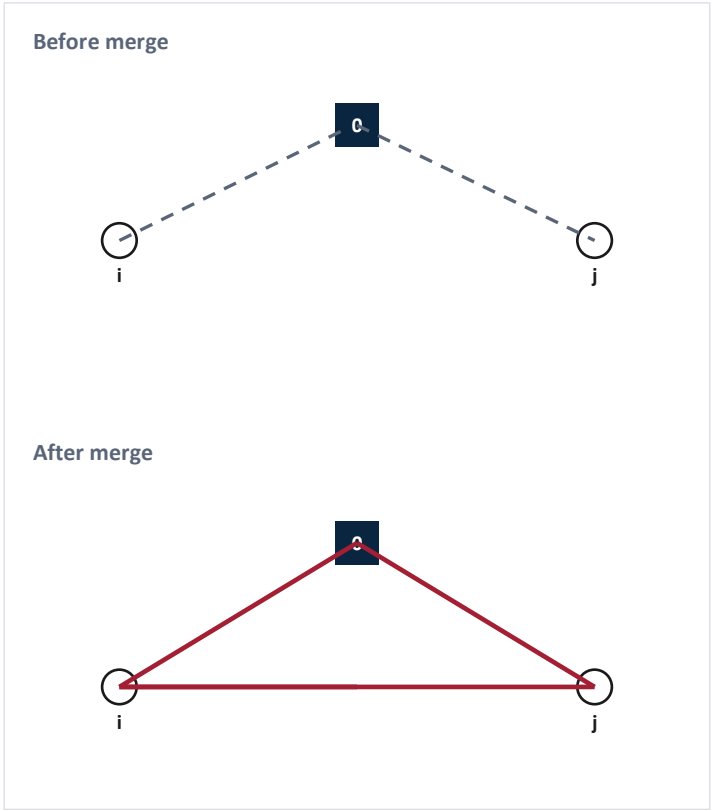
Merging two one-customer routes into one two-customer route saves distance.

$$s(i, j) = c(0, i) + c(0, j) - c(i, j)$$

Savings *from merging routes that serve i and j separately*

Algorithm

1. Start: one route 0 → i → 0 per customer i
2. Compute s(i, j) for every pair (i, j); sort in decreasing order
3. Walk the list: merge the routes containing i and j if feasible (capacity, endpoints)
4. Stop when no merges remain

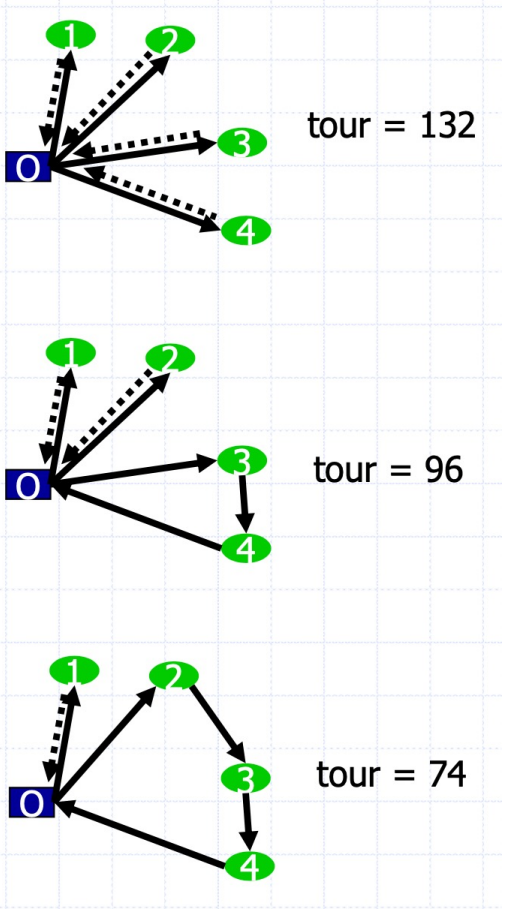


Clarke-Wright Savings (CW)

- Suppose Max Capacity = 3
- Savings = $c_{0i} + c_{j0} - c_{ij}$
  - S(1,2) = 10 + 15 – 8 = 17
  - S(1,3) = 10 + 19 – 23 = 6
  - S(1,4) = 10 + 22 – 35 = -3
  - S(2,3) = 15 + 19 – 12 = 22
  - S(2,4) = 15 + 22 – 21 = 16
  - S(3,4) = 19 + 22 – 5 = 36

Shortest Path Matrix

i\j	0	1	2	3	4
0		10	15	19	22
1			8	23	35
2				12	21
3					5



Local search and neighborhoods

Local search = hill climbing on a neighborhood

- Start from a feasible route set, explore nearby solutions, and move if the objective improves.

Key ideas

- Neighborhood = set of moves reachable in one step
- Hill climbing accepts improving moves
- A local optimum may still be far from the global optimum

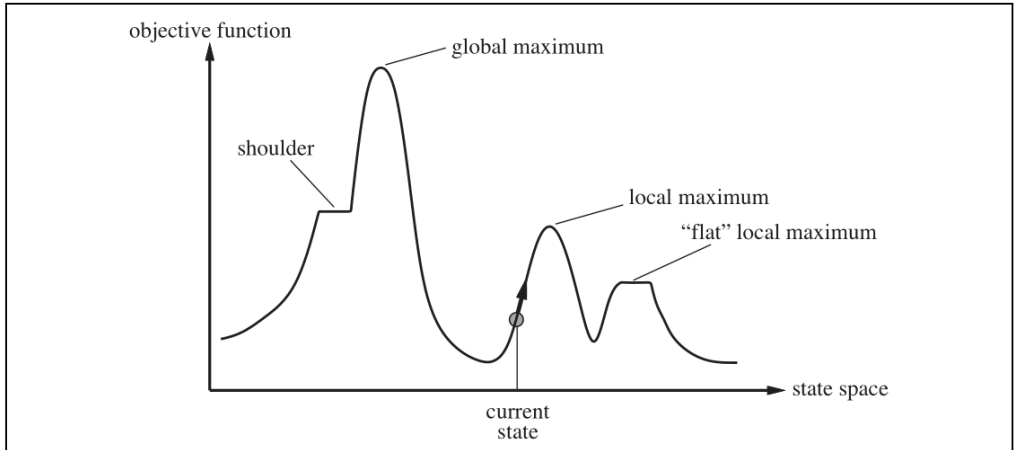
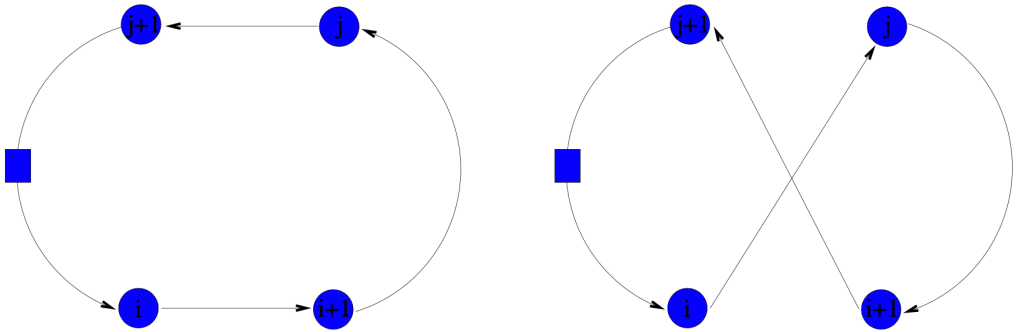


Figure 4.1 A one-dimensional state-space landscape in which elevation corresponds to the objective function. The aim is to find the global maximum. Hill-climbing search modifies the current state to try to improve it, as shown by the arrow. The various topographic features are defined in the text.

Intra-route neighborhoods

2-opt

$$\{i, i+1\}\{j, j+1\} \longrightarrow \{i, j\}\{i+1, j+1\}$$



$O(n^2)$ possible exchanges
One path is reversed

Boundary exchange: drop (i,i+1),(j,j+1); add (i,j),(i+1,j+1)

Gain: $\Delta_2 = c_{i,j} + c_{i+1,j+1} - c_{i,i+1} - c_{j,j+1}$

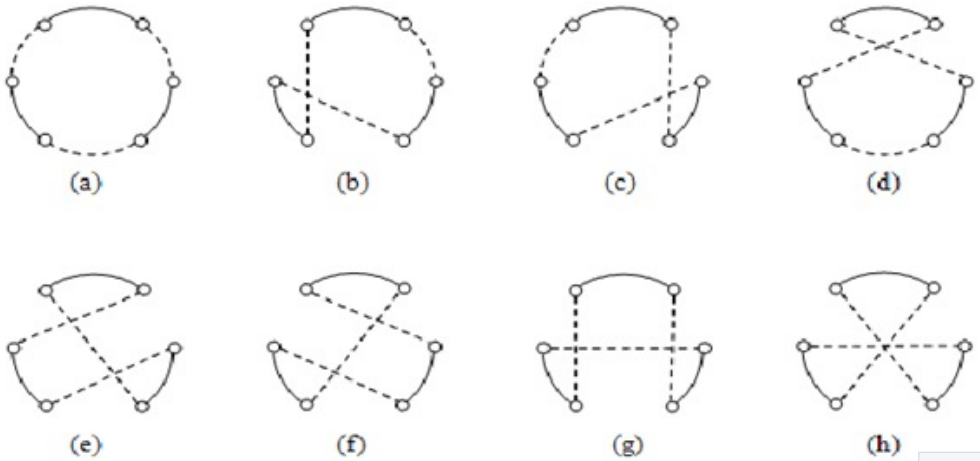
Accept if $\Delta_2 < 0$. In the symmetric case, only the middle path is reversed.

Explainer by Rudi dev (YouTube)

Intra-route neighborhoods

3-opt

$$\{i, i+1\}\{j, j+1\}\{k, k+1\} \longrightarrow \dots$$



$O(n^3)$ possible exchanges
Paths can be reversed

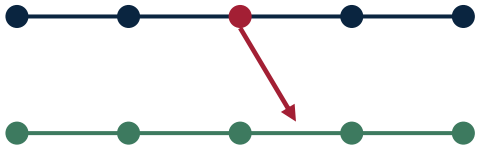
Remove three edges: (i,i+1), (j,j+1), (k,k+1).
Example reconnection A: add (i,j), (i+1,k), (j+1,k+1).
Gain: $\Delta_3^A = [c_{i,j} + c_{i+1,k} + c_{j+1,k+1}] - [c_{i,i+1} + c_{j,j+1} + c_{k,k+1}]$.
Evaluate several reconnections and accept the best one with $\Delta < 0$.

Explainer by Rudi dev (YouTube)

Inter-route neighborhoods

Moves between routes — the core of multi-vehicle local search.

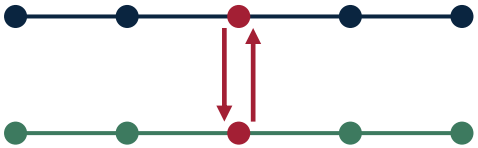
Relocate



Move one customer from its current route to another position (same or different route).

Cheapest to evaluate; simplest operator.

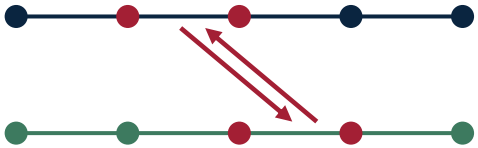
Exchange (swap)



Swap two customers — one from each of two routes — between their positions.

Useful when two routes are near-balanced.

Cross-exchange



Swap entire segments between two routes, preserving the ordering within each segment.

Stronger move; explores larger neighborhood.

Savelsbergh (1992), ORSA Journal on Computing.

Metaheuristics

Local optima

Slide Puzzle Game

3	6	8
5	1	7
2	4	

Solve the puzzle.

Source: Rahul Aware

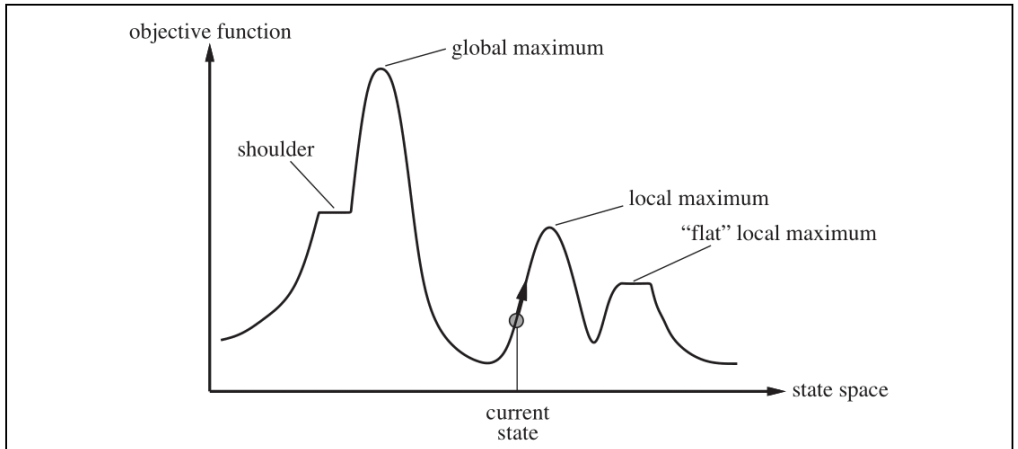


Figure 4.1 A one-dimensional state-space landscape in which elevation corresponds to the objective function. The aim is to find the global maximum. Hill-climbing search modifies the current state to try to improve it, as shown by the arrow. The various topographic features are defined in the text.

Metaheuristics

Higher-level strategies that orchestrate local search to escape local optima.

Intensification — refine the current best; **Diversification** — escape to unexplored regions. Every metaheuristic is a particular way to trade these off.

Simulated annealing

Accept worse moves with probability that decays over time.

Cools from exploration to exploitation.

Tabu search

Keep a short-term memory of recent moves; forbid reversals.

Forces the search out of local minima.

Genetic algorithms

Maintain a population; combine (crossover) and perturb (mutation).

Diversification via population diversity.

Large Neighborhood Search

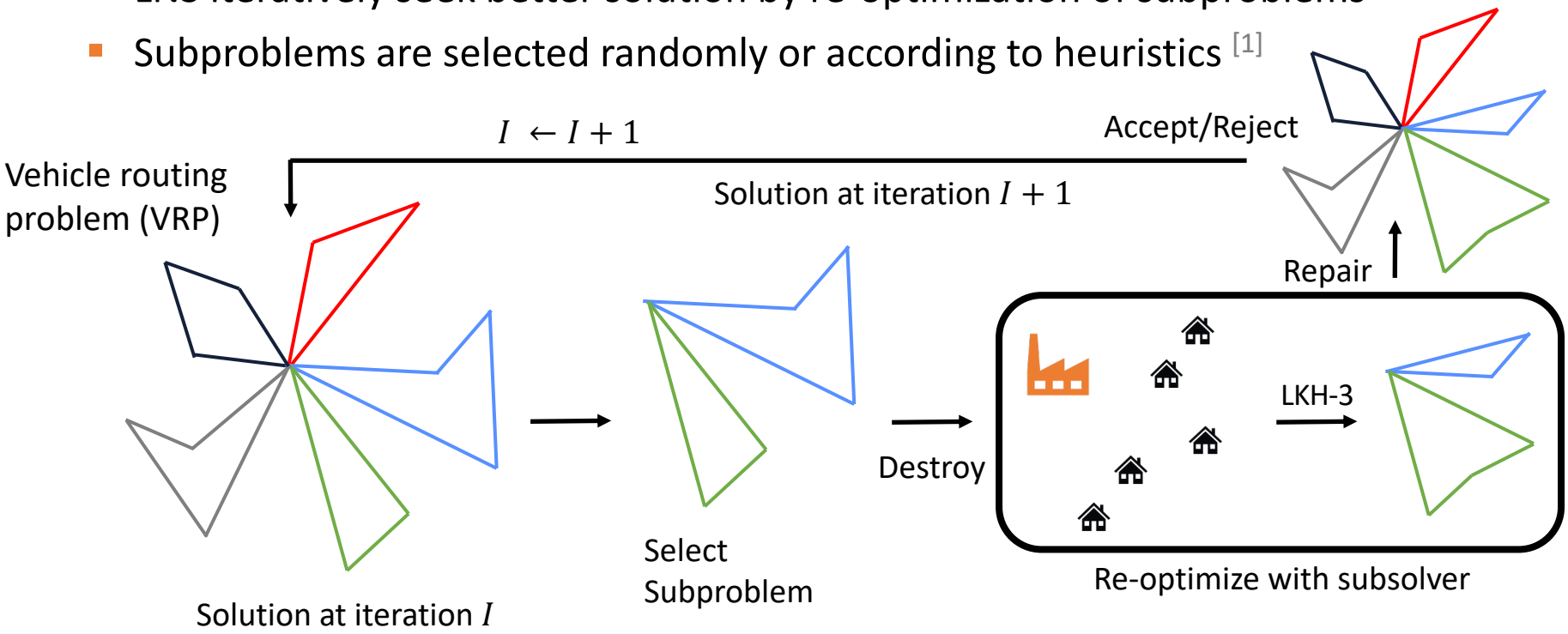
Destroy part of the solution, repair it with a subsolver; repeat.

Today's workhorse for large VRP — next slide.

Modern VRP solvers often combine several layers: construction heuristic → local search → metaheuristic wrapper.

Large Neighborhood Search (LNS)

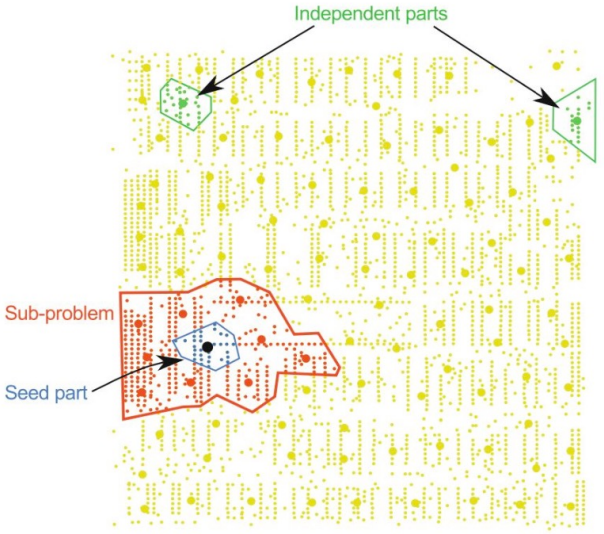
- Large problems often decompose into smaller problems
- LNS iteratively seek better solution by re-optimization of subproblems [1]
- Subproblems are selected randomly or according to heuristics [1]



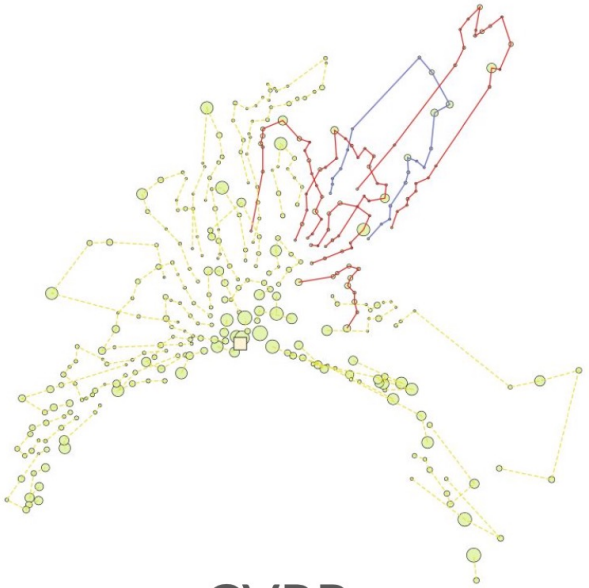
[1] Taillard et al. **Partial OPtimization Metaheuristic Under Special Intensification Conditions (POPMUSIC)**. Springer, 2002

Which subproblem to choose?

Spatial Locality



Clustering



CVRP

Which subproblem to choose?

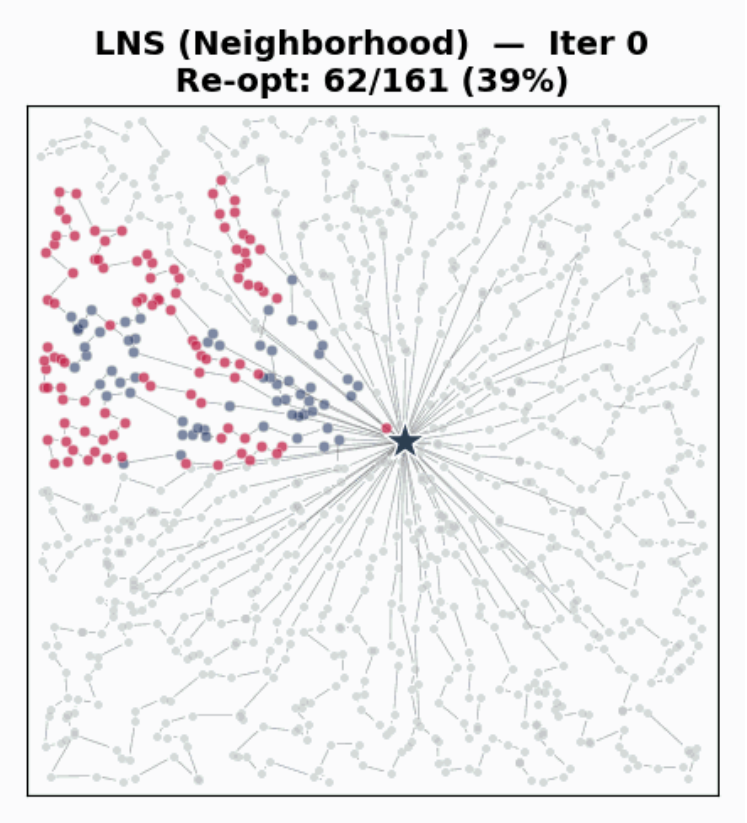
The Restricted Subproblem Selection Space

Without restriction: exponential ways to construct a subproblem	N nodes, R routes $O(2^N)$
Simplification 1: each subproblem has **k** routes	R choose k = $O(R^k)$
Simplification 2: each subproblem is centered around a particular route r and contains the **k**-NN routes	$O(R)$

Which subproblem to choose?

How to choose from among the subproblems?

- **Diversification**
  - Uniformly random
  - Count-based
  - Max Min Distance
- **Intensification**
  - Use a stack (last-in-first-out) [Alvim et al. 2013]



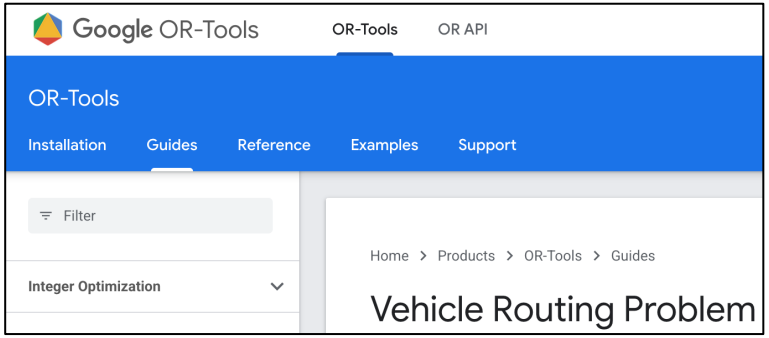
What is the subsolver?

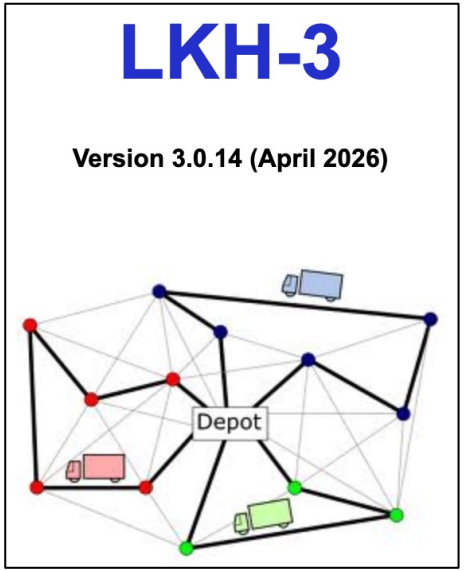
Any valid VRP solver

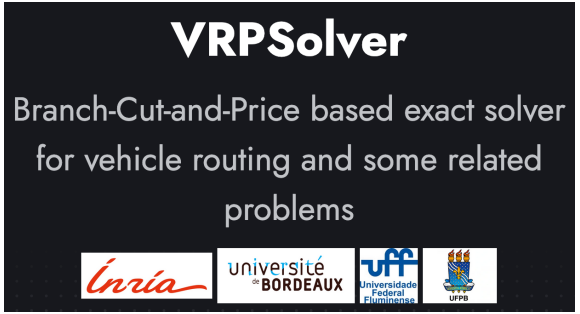
- Heuristics like CW, k-Opt, etc.
- Exact solver for MILP like Gurobi

Popular VRP solvers

- VRPSolver (branch-and-price)
- Google OR-Tools (heuristic)
- LKH-3 (heuristic)
- PyVRP (heuristic – HGS)
- cuOpt (heuristic, GPU-accelerated)







PyVRP

Summary of Routing Methods

	Method	N	Cost	Optimal	Runtime
0	Exact ILP (Gurobi)	20	972.0	Yes	0.937729
1	Clarke-Wright Savings	20	1009.0	No	0.000297
2	Exact ILP (Gurobi)	30	1257.0	No	30.130327
3	Clarke-Wright Savings	30	1284.0	No	0.000683
4	Clarke-Wright Savings	1000	5725.0	No	5.488914
5	CW + 2-Opt Local Search	1000	5685.0	No	5.494523
6	LNS, 6s	1000	5617.0	No	6.000000
7	LNS, 60s	1000	5586.0	No	60.000000
8	HGS, 6s	1000	5579.0	No	6.000000
9	HGS, 60s	1000	5298.0	No	60.000000

Solution methodology across transportation problems

Problem	Construction	Local search	Metaheuristic	SotA solver / ref
CVRP	Clarke-Wright, sweep, insertion	2-opt, 3-opt, relocate, swap	LNS, HGS, tabu	HGS-CVRP, LKH-3, PyVRP
Gate assignment	Greedy by time or slack	Reassign, swap, rotate chain	Tabu, SA, VNS	Dorndorf et al. (2008)
Landing scheduling	FCFS / target-time order	Swap, reinsert, shift	SA, tabu, GRASP, VNS	Beasley et al. (2000)
Berth allocation	Earliest-feasible start	Swap, shift, reinsert	Tabu, SA, adaptive LNS	Bierwirth & Meisel (2015)
Transit design	Greedy demand coverage	Add / drop / swap lines	GA, memetic, SA	Guihaire & Hao (2008)
Signal timing	Webster splits and cycle	Perturb cycle, splits, offset	GA, SA, deep RL	TRANSYT, learned controllers

These problems follow the same construction → local search → metaheuristic progression; SotA solvers are almost always a hybrid.

Advanced vehicle routing

State-of-the-art VRP solver: Hybrid Genetic Search (HGS)

State-of-the-art VRP solvers rarely rely on one idea alone: HGS combines population search, local search, and adaptive constraint handling.

Population
maintain many complete solutions

Crossover
recombine useful route structure

Education
local search improves each offspring

Penalties
allow temporary infeasibility

Diversity
avoid premature convergence

Ingredients

- parent selection from a population
- route-based or giant-tour crossover
- local-search “education” using 2-opt / relocate / swap
- survivor selection based on quality and diversity

Why it works

Intensification from local search + penalty-guided repair.

Diversification from crossover and diversity management.

Net effect: HGS is far stronger than a plain genetic algorithm or a single hill-climber.

Vidal et al. (2013) is the classic HGS reference; modern descendants include HGS-CVRP and many hybrid VRP solvers.

Real-world issues

- The real world does not behave according to uniform assumptions
  - Dock configuration
  - Dock hours
  - Trailer types
  - Moveable bulkheads (bulk liquids, grocery reefers)
  - Truck types
  - Truck-trailer combos: doubles & triples (pups)
  - Compatibility: order-vehicle, order-order, vehicle-site
  - Preferred customers (big box)
  - Driver preferences (seniority, local knowledge)
  - Driver skills (service technician)
  - Rush hour traffic
  - Real-time dispatching (deployed vehicles)
  - Refueling
  - Maintenance

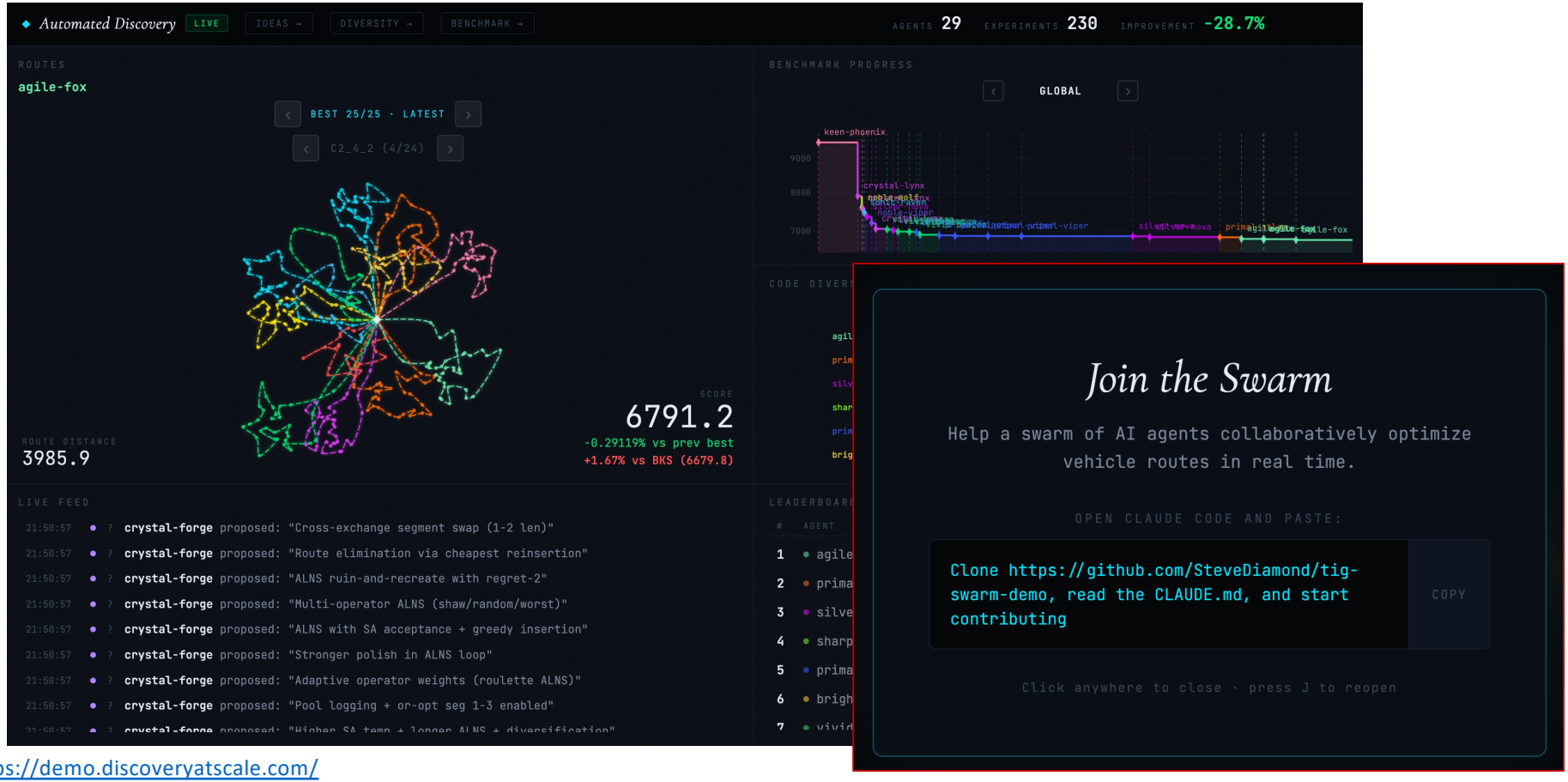
Adapted from Goentzel 2004

Element Interactions

- Truck & Trailer
  - Trailers the tractor can handle – length, pups, specialized (e.g. car hauler)
- Vehicle & Customer
  - Must be able to visit the customer (loading dock, cornering, parking)
- Vehicle & Order
  - Products may not be deliverable on certain resources -- HazMat, loading/handling equipment (tanks, racks), capabilities (refrigeration), physical dimensions, etc.
- Vehicle & Driver
  - Not licensed for the truck, not able to load/unload trailer
- Order & Order
  - Products may not mix (lumber & light bulbs, bottled water & dehydrated food, etc.)

Adapted from Goentzel 2004

Optimizing VRPs with Claude Code



https://demo.discoveryatscale.com/

References

1. Eduardo Uchoa, “Exact Algorithms for Vehicle Routing: advances, challenges, and perspectives” (2024). [pdf]
2. Chris Caplice, “Transportation Management: Vehicle Routing.” ESD260 Logistics Systems (2006). [pdf]

Additional large-scale transportation problems

Airport gate assignment problem

Objective: Minimize total gate assignment cost

$$\text{Minimize} \sum_{f \in F} \sum_{g \in G} c_{fg} x_{fg}$$

subject to:

$$\sum_{g \in G} x_{fg} = 1 \quad \forall f \in F$$

$$x_{fg} + x_{hg} \leq 1 \quad \forall g \in G, \forall (f,h) \in \mathcal{O}$$

Notation

- $F = \{0,1,\dots,n\}$ flights
- $G = \{0,1,\dots,m\}$ gates
- $x_{fg} \in \{0,1\}$, 1 if flight f is assigned to gate g
- c_{fg}: cost of assigning flight f to gate g
- $\mathcal{O} = \{(f,h) \in F \times F : f < h$, flights f and h overlap in time$\}$: conflict set of flights

Degree Constraints: Each flight is assigned a gate

Conflict Constraints: Overlapping flights cannot share a gate

Why heuristics / metaheuristics are common

Time conflicts

Overlapping flights sharing a gate create many pairwise incompatibilities.

Rich compatibility

Aircraft size, domestic/international rules, towing options, and walking-distance penalties complicate the model.

Natural neighborhood moves

Move one flight to a new gate; swap two flights; rotate a short conflict chain.

Typical scalable approach

Greedy construction → local reassignment / swap search → tabu, simulated annealing, or variable NS.

Aircraft landing scheduling problem

Objective: Minimize total earliness / lateness penalty

$$\text{Minimize} \sum_{i \in A} \alpha_i e_i + \beta_i \ell_i$$

subject to:

$$t_i \geq t_j + s_{ij} - M(1 - y_{ij}) \quad \forall i \neq j$$

$$y_{ij} + y_{ji} = 1 \quad \forall i, j \in A, i \neq j$$

$$E_i \leq t_i \leq L_i \quad \forall i \in A$$

Notation

- $A = \{1, \ldots, n\}$ aircraft
- t_i: landing time of aircraft *i*
- $y_{ij} \in \{0,1\}$: 1 if *i* lands before *j*
- s_{ij}: required separation *i* → *j*
- $[Ei, Li]$: time window; e_i, ℓ_i earliness, lateness; penalties α_i, β_i

Separation constraints: minimum gap between successive landings

Precedence constraints: every pair has a fixed landing order

Window constraints: each landing within its feasible window

Why heuristics / metaheuristics are common

Ordering decisions

Sequencing n aircraft already induces O(n²) pairwise precedence choices.

Propagation

A small delay early in the sequence can ripple through all later separations.

Natural neighborhood moves

Swap two aircraft, reinsert one aircraft elsewhere, or shift landing times within a fixed sequence.

Typical scalable approach

FCFS / greedy-by-target-time start → swap/reinsert local search → tabu, SA, GRASP, or VNS.